

Oracle® Essbase

Technical Reference

RELEASE 11.1.2.1

ORACLE®

**ENTERPRISE PERFORMANCE
MANAGEMENT SYSTEM**

Essbase Technical Reference, 11.1.2.1

Copyright © 1996, 2011, Oracle and/or its affiliates. All rights reserved.

Authors: EPM Information Development Team

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited. The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS:

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Documentation Accessibility	15
Chapter 1. Oracle Essbase Technical Reference Overview	17
<i>About the Oracle Essbase Technical Reference</i>	17
What You Should Know Before You Start	17
Sample Applications	17
Syntax Conventions	18
About Aggregate Storage Databases	19
Chapter 2. Calculation Functions	21
Calculation Functions Overview	21
Generations and Levels	21
Abbreviations	22
Function Syntax	22
Function Parameters	23
Calculation Operators	24
Mathematical Operators	24
Conditional and Logical Operators	25
Cross-Dimensional Operators	25
Operation Results on #MISSING Values and Zero (0) Values	25
Calculation Function Categories	27
Boolean Functions	27
Relationship Functions	28
Mathematical Functions	29
Member Set Functions	30
Range and Financial Functions	33
Allocation Functions	35
Forecasting Functions	35
Statistical Functions	36
Date & Time Function	36
Miscellaneous Functions	37
Custom-defined Functions	37

Calculation Function Reference	37
Custom-Defined Calculation Functions	234
Java Code Examples	234
MaxL Registration Scripts	264
Custom-Defined Macros	292
Custom-Defined Macro Input Parameters	292
Using Argument Values in Macro Definitions	294
Directives Used in Custom-Defined Macros	295
Macro Reference	295
Chapter 3. Calculation Commands	303
Calculation Commands Overview	303
Calculation Operators	303
Mathematical Operators	304
Conditional and Logical Operators	304
Cross-Dimensional Operator	305
Calculation Command Groups	305
Conditional Commands	305
Control Flow Commands	306
Data Declaration Commands	306
Functional Commands	306
Member Formulas	307
Calculation Command Reference	308
Chapter 4. Essbase.cfg Configuration Settings	369
Configuration File Overview	369
Configuring Essbase.cfg	369
Essbase.cfg Setting Categorical List	370
Backup and Recovery Configuration Settings	371
Calculation Configuration Settings	371
Data Import and Export Configuration Settings	372
Hybrid Analysis Configuration Settings	372
Failover Clustering Configuration Settings	372
Logging and Error Handling Configuration Settings	373
Memory Management Configuration Settings	374
Miscellaneous Configuration Settings	374
Partitioning Configuration Settings	374
Ports and Connections Configuration Settings	375
Query Management Configuration Settings	375
Security File Configuration Settings	376

SSL Configuration Settings	376
Aggregate Storage and Block Storage Settings Comparison	376
Configuration Settings Reference	380
AGENTDELAY	382
AGENTDESC	383
AGENTDISPLAYMESSAGELEVEL	383
AGENTLEASEEXPIRATIONTIME	384
AGENTLEASEMAXRETRYCOUNT	385
AGENTLEASERENEWALTIME	385
AGENTLOGMESSAGELEVEL	386
AGENTPORT	387
AGENTSECUREPORT	388
AGENTTHREADS	388
AGGRESSIVEBLKOPTIMIZATION	389
AGTMAXLOGFILESIZE	390
AGTSVRCONNECTIONS	391
APPMAXLOGFILESIZE	392
APSRESOLVER	392
ASOLOADBUFFERWAIT	393
ASOSAMPLESIZEPERCENT	394
AUTHENTICATIONMODULE	395
CALCCACHE	396
CALCCACHEHIGH	397
CALCCACHEDEFAULT	398
CALCCACHELOW	399
CALCLIMITFORMULARECURSION	400
CALCLOCKBLOCK	401
CALCMODE	402
CALCNOTICE	403
CALCOPTFRMLBOTTOMUP	404
CALCREUSEDYNCALCBLOCKS	405
CALCPARALLEL	406
CALCTASKDIMS	407
CCTRACK	408
CLEARLOGFILE	410
CLIENTPREFERREDMODE	411
CRASHDUMP	411
DATACACHE SIZE	412
DATAERRORLIMIT	413

DATAEXPORTENABLEBATCHINSERT	414
DATAFILECACHESIZE	415
DEFAULTLOGLOCATION	415
DELAYEDRECOVERY	416
DELIMITEDMSG	417
DELIMITER	417
DEXPSQLROWSIZE	418
DIMBUILDERRORLIMIT	419
DIMBUILDSTATSINTERVAL	420
DIRECTIO	420
DISABLEREPLMISSINGDATA	421
DISKVOLUMES	422
DISPLAYMESSAGELEVEL	423
DLSINGLETHREADPERSTAGE	424
DLTHREADSPREPARE	426
DLTHREADSWRITE	427
DYNALCCACHEBLKRELEASE	429
DYNALCCACHEBLKTIMEOUT	430
DYNALCCACHECOMPRBLKBUFSIZE	432
DYNALCCACHEMAXSIZE	433
DYNALCCACHEONLY	435
DYNALCCACHEWAITFORBLK	436
ENABLE_DIAG_TRANSPARENT_PARTITION	438
ENABLECLEARMODE	439
ENABLESECUREMODE	440
ENABLESWITCHTOBACKUPFILE	440
ESSBASEFAILOVERTRACELEVEL	441
ESSBASESERVERHOSTNAME	442
EXCEPTIONLOGOVERWRITE	442
EXCLUSIVECALC	444
EXPORTTHREADS	444
FAILOVERMODE	445
FILELOCKINGMODE	446
FORCEALLDENSECALCON2PASSACCOUNTS	446
FORCEGRIDEXPANSION	447
GRIDEXPANSION	448
GRIDEXPANSIONMESSAGES	448
HAENABLE	449
HAMAXNUMCONNECTION	449

HAMAXNUMSQLQUERY	450
HAMAXQUERYROWS	451
HAMAXQUERYTIME	452
HAMEMORYCACHESIZE	453
HARAGGEDHIERARCHY	454
HARETRIEVENUMROW	455
HASOURCEDSNOS390	456
HISLEVELDRILLTHROUGH	457
IBHFIXTHRESHOLD	457
IDMIGRATION	459
IMPLIED_SHARE	459
INCRESTRUC	460
INDEXCACHESIZE	463
JVMMODULELOCATION	463
LOCKTIMEOUT	464
LOGINFAILUREMESSAGEDETAILED	465
LOGMESSAGELEVEL	465
LROONSHAREDMBR	466
MAXERRORMBRVERIFYREPORT	467
MAXFORMULACACHESIZE	467
MAXLOGINS	468
MAX_REQUEST_GRID_SIZE	468
MAX_RESPONSE_GRID_SIZE	469
MDXFORMULARECURSIONLIMIT	470
MEMSCALINGFACTOR	471
MULTIPLEBITMAPMEMCHECK	472
NETBINDRETRYDELAY	473
NETDELAY	473
NETRETRYCOUNT	474
NETSSLHANDSHAKETIMEOUT	474
NETTCPCONNECTRETRYCOUNT	475
NOMSGLOGGINGONDATAERRORLIMIT	476
NUMBEROFSECFILEBACKUPS	476
NUMERICPRECISION	477
OUTLINECHANGELOG	478
OUTLINECHANGELOGFILESIZE	479
PARCALCMULTIPLEBITMAPMEMOPT	479
PERSISTUSERATLOGIN	480
PIPEBUFFERSIZE	480

PORTINC	481
PORTUSAGELOGINTERVAL	482
PRELOADALIASNAMESPACE	483
PRELOADMEMBERNAMESPACE	483
PRELOADUDANAMESPACE	484
QRYGOVEXECBLK	485
QRYGOVEXECTIME	486
REPLAYSECURITYOPTION	488
REPLICATIONASSUMEIDENTICALOUTLINE	489
RTDEPCALCOPTIMIZE	490
SECFILEBACKUPINTERVAL	490
SECURITYFILECOMPACTIONPERCENT	491
SERVERLEASEEXPIRATIONTIME	492
SERVERLEASEMAXRETRYCOUNT	492
SERVERLEASERENEWALTIME	493
SERVERPORTBEGIN	493
SERVERPORTEND	494
SERVERTHREADS	496
SILENTOTLQUERY	497
SPLITARCHIVEFILE	497
SQLFETCHERRORPOPUP	498
SSAUDIT	499
SSAUDITR	500
SSINVALIDTEXTDETECTION	501
SSLCIPHERSUITES	502
SSLOGUNKNOWN	503
SSOPTIMIZEDGRIDPROCESSING	503
SSPROCROWLIMIT	504
SUPNA	505
TARGETASOOPT	506
TARGETTIMESERIESOPT	507
TIMINGMESSAGES	507
TRANSACTIONLOGDATALOADARCHIVE	508
TRANSACTIONLOGLOCATION	510
TRIGMAXMEMSIZE	511
UNICODEAGENTLOG	512
UPDATECALC	513
VLBREPORT	514
WALLETPATH	515

XOLAPENABLEHEURISTICS	516
XOLAPMAXNUMCONNECTION	516
XOLAPSCHEMAVERIFICATION	517
XOLAPSQLIDLEPERIOD	518
Chapter 5. ESSCMD Commands	519
ESSCMD Overview	519
ESSCMD Getting Started	519
Starting ESSCMD	520
Canceling ESSCMD Operations	520
Quitting ESSCMD	520
ESSCMD Syntax Guidelines	520
Quotation Marks in ESSCMD	521
ESSCMD Semicolon Statement Terminator	521
Referencing Files	521
ESSCMD Batch Processing	522
Writing Script Files	523
Running Script Files	523
Handling Command Errors in a Script File	523
Sample Script Files	524
Writing Batch Files	525
Handling Command Errors in Batch Files	525
ESSCMD Interactive Mode	526
Logging On to Essbase Server	527
Entering Commands	527
Canceling Operations	528
ESSCMD Command Groups	528
Using ESSCMD	528
Application and Database Administration	529
User and Group Security	529
Security Filters and Locks	530
Database Objects	530
Outline and Attribute Information	531
Dimension Building	531
Data Loading, Clearing, and Exporting	531
Calculating	531
Reporting	532
Partitioning	532
Outline Synchronization	532

Error and Log Handling	533
Currency Conversion Information	533
Location Aliases	533
Substitution Variables	533
Aliases	534
Integrity, Performance	534
Backing Up	534
ESSCMD Command Reference	534
Chapter 6. MaxL	623
Overview of MaxL and MDX	623
How to Read MaxL Railroad Diagrams	624
Anatomy of MaxL Statements	624
Railroad Diagram Symbols	624
Sample Railroad Diagram	625
MaxL Data Definition Language (DDL)	626
MaxL Statements	626
Performance Statistics in MaxL	627
Listed By Verbs	633
Listed by Objects	636
MaxL Statement Reference	642
MaxL Definitions	767
MaxL Syntax Notes	768
Numbers in MaxL Syntax	769
Terminals	769
Privileges and Roles	813
Quoting and Special Characters Rules for MaxL Language	816
MaxL Shell Commands	818
Overview of MaxL Shell	819
MaxL Shell Invocation	819
MaxL Shell Syntax Rules and Variables	828
MaxL Shell and Unicode	833
MaxL Shell Command Reference	833
MaxL Perl Module	841
Installation Help	841
Functions	843
Perl Scripting Examples	845
ESSCMD Script Conversion	848
ESSCMD Script Utility Usage	848

Things to Note About the ESSCMD Script Utility	848
ESSCMD to MaxL Mapping	849
Reserved Words List	855
MaxL Statements (Aggregate Storage)	864
Outline Paging Dimension Statistics	904
Aggregate Storage Runtime Statistics	905
MaxL Statements for Data Mining	907
Data Mining Algorithms	907
Data Mining Transformations	907
Mining Models	907
Mining Results	908
Mining Task Templates	908
Mining Sessions	908
Data Mining Statements Listed by Verbs	908
Data Mining Statements Listed by Objects	909
Data Mining Statements	910
MaxL Use Cases	920
Creating an Aggregate Storage Sample Using MaxL	920
Loading Data Using Buffers	921
Using Aggregate Storage Data Load Buffers	923
Specifying Port Numbers in Partition Host Names	923
Using Host Name Aliases When Partitioning	924
Partitioning and SSL	925
Forcing Deletion of Partitions	925
Metadata Filtering	926
Examples of Triggers	927
Chapter 7. MDX	931
Overview of MDX	931
MDX Query Format	932
MDX Syntax and Grammar Rules	932
Understanding BNF Notation	933
MDX Grammar Rules	934
MDX Syntax for Specifying Duplicate Member Names and Aliases	947
MDX Axis Specifications	949
MDX Slicer Specification	952
MDX Cube Specification	952
MDX Set Specification	953
MDX With Section	954

MDX Dimension Specification	958
MDX Layer Specification	959
MDX Member Specification	960
MDX Hierarchy Specification	961
MDX Tuple Specification	961
MDX Create Set / Delete Set	962
MDX Operators	963
About MDX Properties	964
MDX Intrinsic Properties	965
MDX Custom Properties	965
MDX Optimization Properties	966
Querying for Member Properties in MDX	968
The Value Type of MDX Properties	969
MDX NULL Property Values	969
MDX Comments	970
MDX Query Limits	971
Aggregate Storage and MDX Outline Formulas	973
MDX Functions	989
MDX Functions that Return a Member	990
MDX Functions that Return a Set	991
MDX Functions that Return a Tuple	993
MDX Functions that Return a Number	993
MDX Functions that Return a Dimension	995
MDX Functions that Return a Layer	995
MDX Functions that Return a Boolean	995
MDX Functions that Return a Date	996
MDX Functions that Return a String	996
MDX Function Reference	997
Chapter 8. Query Logging Configuration	1161
Query Logging Overview	1161
Query Logging Settings Procedure	1161
Query Log Settings File Syntax	1162
Query Logging Sample File	1165
Query Logging Sample Output	1165
Chapter 9. Report Writer Commands	1169
Report Writer Overview	1169
Report Writer Syntax	1170
Report Delimiters	1170

Syntax Guidelines	1170
Referencing Static Members	1171
Report Writer Command Groups	1171
Report Layout Commands	1172
Data Range Commands	1172
Data Ordering Commands	1172
Member Selection and Sorting Commands	1172
Format Commands	1173
Column or Row Calculation Commands	1176
Member Names and Aliases	1176
Examples of Report Scripts	1177
Sample 1: Creating a Different Format for Each Page	1178
Sample 2: Handling Missing Values	1179
Sample 3: Nesting Columns	1181
Sample 4: Grouping Rows	1182
Sample 5: Reporting on Different Combinations of Data	1186
Sample 6: Formatting Different Combinations of Data	1187
Sample 7: Using Aliases	1189
Sample 8: Creating Custom Headings and % Characters	1190
Sample 9: Creating Custom Page Headings	1193
Sample 10: Using Formulas	1195
Sample 11: Placing Two-Page Layouts on the Same Page	1196
Sample 12: Formatting for Data Export	1198
Sample 13: Creating Asymmetric Columns	1199
Sample 14: Calculating Columns	1200
Sample 15: Calculating Rows	1202
Sample 16: Sorting by Top or Bottom Data Values	1207
Sample 17: Restricting Rows	1209
Sample 18: Ordering Data Values	1210
Sample 19: Narrowing Member Selection Criteria	1211
Sample 20: Using Attributes in Member Selection	1212
Sample 21: Using the WITHATTR Command in Member Selection	1213
Report Writer Command Reference	1214
Chapter 10. Essbase Unicode File Utility	1377
Essbase Unicode File Utility Overview	1377
Types of Encoding Indicators	1378
Determining Whether to Use UTF-8 or Non-Unicode Text Files	1378
When to Use the Essbase Unicode File Utility	1379

Essbase Unicode File Utility Syntax	1379
Index	1383

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at <http://www.oracle.com/accessibility/>.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Access to Oracle Support for Hearing-Impaired Customers

Oracle customers have access to electronic support through My Oracle Support or by calling Oracle Support at 1.800.223.1711. Hearing-impaired customers in the U.S. who wish to speak to an Oracle Support representative may use a telecommunications relay service (TRS). Information about the TRS is available at <http://www.fcc.gov/cgb/consumerfacts/trs.html/>, and a list of telephone numbers is available at <http://www.fcc.gov/cgb/dro/trsphonebk.html>. International hearing-impaired customers should use the TRS at +1.605.224.1837. An Oracle Support engineer will respond to technical issues according to the standard service request process.

1

Oracle Essbase Technical Reference Overview

In This Chapter

About the Oracle Essbase Technical Reference	17
About Aggregate Storage Databases.....	19

About the Oracle Essbase Technical Reference

The *Oracle Essbase Technical Reference* describes commands, functions, and configuration aspects of Oracle Essbase. This reference is intended for advanced users who need detailed information and examples about Essbase elements.

For other information about Essbase, see the *Oracle Essbase Database Administrator's Guide*.

- [“What You Should Know Before You Start” on page 17](#)
- [“Sample Applications” on page 17](#)
- [“Syntax Conventions” on page 18](#)

What You Should Know Before You Start

To use this document, you need the following:

- A working knowledge of the operating system your server uses and the ones your clients use.
- An understanding of Essbase concepts and features.
- An understanding of the typical database administration requirements and tasks, including calculation, reporting, security, and maintenance.

Sample Applications

This document provides examples based on the Sample and Demo applications provided with Essbase. The Sample application contains three databases: Basic, Interntl, and Xchgrate. The Demo application contains one database: Basic. If, when you connect to the Essbase Server, any of the following problems occur, contact your administrator.

- You cannot find the Sample or Demo application
- You don't have adequate access to the Sample or Demo application
- You don't see any data in the Sample or Demo databases

Syntax Conventions

This document uses several formatting styles to indicate actions you should take or types of information you need.

Syntax	Purpose	Example
UPPERCASE	Command or function names in syntax.	BEGINARCHIVE
<i>italic</i>	Terms, such as parameters, that you replace with a value	ESSGETSTRING (<i>hCtx</i> , <i>pString</i>); <i>\directorypath\filename</i> The <i>dimList</i> argument...
" "	Double quotation marks enclose text parameters or single parameters that include a space	" <i>appName</i> " SETDEFAULTCALC "CALC ALL";
!	Report Writer: The report output character (bang) signals the start of report processing; this character must be on its own line	... // commands <DESC MARKET !
()	Parentheses are used in a couple of ways: <ul style="list-style-type: none"> ● To enclose function parameters ● To show the order of execution of the enclosed operations 	ESSGETSTRING (<i>hCtx</i> , <i>pString</i>); (<i>a + b</i>) * <i>c</i>
//	Comment markers in report scripts. Double slashes (//) indicate text from // to end of line should be ignored in processing.	// Get results
/* ... */	Comment markers in calculation scripts. The /* ... */ comment markers indicate the enclosed text should be ignored in processing.	/*Get results*/
;	Statement terminator	EXIT;
[]	Brackets enclose optional parameters in syntax . Used with OR symbol if there is more than one optional parameter. Do not type brackets or the OR symbol .	INDENT [<i>offset</i>]
[, <i>numeric</i>] [, "text"]	Indicates an optional numeric (no quotes) or character (quoted) parameter and the comma which must precede the optional parameter. Do not type the brackets.	[, <i>year</i>] [, " <i>columnName</i> "]
{ }	Braces group statements for processing, enclose alternatives, one of which you must choose Report Writer: Enclose report formatting commands	HELP { ? <i>commandName</i> } { SUPFORMATS }
	Syntax: OR. Separates alternatives from which you choose only one. Do not type the OR symbol.	SET AGGMISSG ON OFF
<	Report Writer: Angle bracket precedes layout and member selection commands.	<PAGE
@	Essbase calculation functions: Precedes many function names	@ABS

Syntax	Purpose	Example
->	Essbase calculation functions: Cross-dimensional operator (a hyphen followed by a greater-than sign) points to data values of specific member combinations -> (cross-dimensional operator)	Price -> West = AVGRANGE

About Aggregate Storage Databases

This topic explains how the elements discussed in the *Oracle Essbase Technical Reference* apply to aggregate storage databases.

Consider using the aggregate storage storage model if the following is true for your database:

- The database is sparse and has many dimensions, and/or the dimensions have many levels of members.
- The database is used primarily for read-only purposes, with few or no data updates.
- The outline contains no formulas except in the dimension tagged as Accounts.
- Calculation of the database is frequent, is based mainly on summation of the data, and does not rely on calculation scripts.

Note the applicability of the following elements for aggregate storage databases:

- **MDX**—Used for querying on block storage and aggregate storage databases. Additionally, MDX numeric-value expressions can be used for developing formulas on aggregate storage outlines. For more information, see [“Aggregate Storage and MDX Outline Formulas” on page 973](#).
- **Calculation commands**—Not supported in enterprise analytics databases, because calculation scripts are not relevant to aggregate storage storage.
- **Calculation functions**—Not supported in enterprise analytics databases. Instead, MDX formulas can be written using MDX numeric-value expressions. Only the Accounts dimension can have formulas in aggregate storage databases.
- **Report Writer commands**—All Report Writer commands (except <SPARSE) are supported for aggregate storage databases.
- **MaxL statements**—Some MaxL grammar is applicable to aggregate storage mode, and some MaxL grammar is not relevant. To learn which statements are supported in aggregate storage application and database operations, see [“MaxL Statements \(Aggregate Storage\)” on page 864](#).
- **ESSBASE.CFG configuration settings**—Some ESSBASE.CFG configuration settings are applicable to aggregate storage mode, and some are not. To learn which settings are supported in aggregate storage mode, see [“Aggregate Storage and Block Storage Settings Comparison” on page 376](#).

For more information about aggregate storage storage, see the *Oracle Essbase Database Administrator's Guide*.

2

Calculation Functions

In This Chapter

Calculation Functions Overview.....	21
Function Syntax	22
Function Parameters.....	23
Calculation Operators	24
Operation Results on #MISSING Values and Zero (0) Values	25
Calculation Function Categories.....	27
Calculation Function Reference.....	37
Custom-Defined Calculation Functions	234
Custom-Defined Macros.....	292

Calculation Functions Overview

Essbase provides a suite of functions and calculation operators to facilitate the definition and application of complex member formulas. Both the Outline Editor and the Calculation Script Editor provide dialog boxes containing functions and operators that you can paste into member formulas and calc scripts. For more information, see the *Oracle Essbase Database Administrator's Guide*.

The topics for individual functions in this section provide examples that are based on an application and database provided with the Essbase Server software, called Sample Basic. If you do not have access to Sample Basic, contact your administrator.

Generations and Levels

Many Essbase functions identify a member in the database by its position in the database outline. The outline structure represents a hierarchical tree; every dimension represents a subsection of the database tree. Generations and levels provide position references for all database members within the tree. Position references are required because many applications must be able to determine the location of members within the database structure.

The terms "generation" and "level" denote the distance from either the "root" or the "leaves" of the dimension. Thus, you can determine the location of any member within a database tree. You can also specify relationships between groups of related members.

Generations specify the distance of members from the root of their dimension. All members in a database that are the same number of branches from their root have the same generation number. The dimension is generation 1, its children are generation 2, and so on.

Levels measure the number of branches between a member and the lowest member below it, that is, the number of branches between a member and the "leaf" of its hierarchy within the database structure. Level 0 specifies the bottom-most members of a dimension and thus provides ready access to the raw data stored in a database. Leaf members are level 0, then their parents are level 1, and so on up the hierarchy.

You might note that when all sibling members have the same generation number but not necessarily the same level number.

For example, the members in this hierarchy:

```
Dim1
  m11
    m111
    m112
  m12
    m121
    m122
  m13
```

have the following generation and level numbers:

```
Dim1      Gen 1, Level 2
  m11      Gen 2, Level 1
    m111    Gen 3, Level 0
    m112    Gen 3, Level 0
  m12      Gen 2, Level 1
    m121    Gen 3, Level 0
    m122    Gen 3, Level 0
  m13      Gen 2, Level 0
```

Abbreviations

Function abbreviations are not supported. Use the full function name to obtain expected behavior.

Function Syntax

The individual topics for each function include the required syntax for that function. Function names appear in **bold**; required parameters appear in *italics*; and optional parameters appear in brackets [] and *italics*. Individual topics also discuss the defaults that are used when optional parameters are not specified. For detailed descriptions of each function, along with examples of usage, please refer to the individual topic.

For information about how Essbase checks for and responds to syntax errors in formulas and calculation scripts, or for information on how to use semicolons in formulas and calculation scripts, see the *Oracle Essbase Database Administrator's Guide*.

Function Parameters

The following table provides a brief description of some of the common parameters used in various functions.

Note: Member names that are also keywords, such as IF, THEN, ELSE, and RETURN, must be enclosed in quotation marks. It is recommended practice to always enclose member names in quotation marks.

Parameter	Description
<i>attDimName</i>	A single attribute dimension name specification. @WITHATTR (Ounces , "<" , 16)
<i>attMbrName</i>	A single attribute member name specification. @ATTRIBUTE (Can) @ATTRIBUTEVAL (Ounces) @WITHATTR (" Pkg Type" , "= =" , Can)
<i>dimName</i>	A single dimension name specification. @CURLEV (Accounts) @CURGEN (Year) @PARENT (Measures , Sales)
<i>explist</i>	A comma-delimited list of member names, variable names, functions, and numeric expressions, all of which return numeric values. @MAX (Jan , Feb , 100 , Apr-May) @MIN (Oct:Dec) @COUNT (SKIPNONE , @RANGE (Sales , @CHILDREN (Product)))
<i>expression</i>	Any mathematical or numeric expression that is valid within Essbase and that, when calculated, returns a numeric value. This definition of <i>expression</i> also includes parameters such as <i>numDigits</i> , <i>generation</i> , and <i>level</i> , and other similar parameters for the financial group of functions, such as <i>rateMbrConst</i> and <i>lifeMbrConst</i> . @ABS (Actual-Budget) @ROUND (Sales / 10.0 + 100)
<i>genLevName</i>	Generation or level name specification. @DESCENDANTS (Market , Regions) @RELATIVE (Qtr1 , Month)
<i>genLevNum</i>	An integer value that defines the number of a generation or level. A positive integer defines a generation number. A value of 0 or a negative integer defines a level number. @ANCESTORS (Sales , -2) @SANCESTVAL (Product , 2 , Sales)
<i>mbrList</i>	A comma-delimited list of members. @ISMBR (New_York , Boston , Chicago)

Parameter	Description
<i>mbrName</i>	<p>Any valid single member name or member combination, or a function that returns a single member or member combination. This definition also includes similar parameters, such as <i>balanceMbrName</i>, <i>costMbr</i>, and <i>cashflowMbr</i>, for the financial group of functions.</p> <p>@GEN (Actual) @CHILDREN (Product) @MAXRANGE (@ANCESTORS (Qtr4) , Jan : Dec)</p> <p>For functions that expect a single member name (for example, @DESCENDANTS and @CHILDREN), if a member combination is provided, Essbase uses the first member in the combination. For example, if <i>mbrName</i> is Utah->Sales, Essbase uses Utah.</p>
<i>n</i>	<p>A positive or negative integer value.</p> <p>@NEXT (2 , Jan : Dec) @SHIFT (3)</p>
<i>rangeList</i>	<p>A valid member name, a comma-delimited list of member names, member set functions, and range functions from the same dimension. If <i>rangeList</i> is optional and is not specified, Essbase uses the level 0 members from the dimension tagged as Time. If no dimension is tagged as Time and this parameter is omitted, Essbase reports a syntax error. This definition of <i>rangeList</i> also includes <i>mbrList</i>.</p> <p>@ACCUM (Q189 : Q491) @MAXRANGE (Sales , @CHILDREN (Qtr1))</p>
<i>tag</i>	<p>Any valid account tag defined in the current database including First, Last, Average, Expense, and Two-Pass.</p> <p>@ISACCTYPE (" EXPENSE ")</p> <p>To ensure that the tag is resolved as a string rather than a member name, it is recommended to enclose it in quotation marks.</p>

Calculation Operators

Calculation operators (mathematical, conditional and logical, and cross-dimensional) define equations for member formulas and calc scripts.

Mathematical Operators

Mathematical operators perform common arithmetic operations.

Operator	Description
+	Adds.
-	Subtracts.
*	Multiplies.
/	Divides.
%	Evaluates percentage. For example, <i>Member1%Member2</i> evaluates <i>Member1</i> as a percentage of <i>Member2</i> .

Operator	Description
()	Controls the order of calculations and nests equations and formulas.

Conditional and Logical Operators

Conditional operators build logical condition into calculations.

Operator	Description
IF ELSE ELSEIF ENDIF	Tests conditions and calculates a formula based on the success or failure of the test.
>	Data value is greater than.
>=	Data value is greater than or equal to.
<	Data value is less than.
<=	Data value is less than or equal to.
= =	Data value is equal to.
< > or !=	Data value is not equal to.
AND	Logical AND linking operator for multiple value tests. Result is TRUE if both conditions are TRUE. Otherwise the result is FALSE.*
OR	Logical OR linking operator for multiple value tests. Result is TRUE if either condition is TRUE. Otherwise the result is FALSE.*
NOT	Logical NOT operator. Result is TRUE if condition is FALSE. Result is FALSE if condition is TRUE.*

* The logical constants TRUE and FALSE are interpreted as 1 (TRUE) and 0 (FALSE) where appropriate.

Cross-Dimensional Operators

The cross-dimensional operator (->) points to data values of specific member combinations.

The cross-dimensional operator is created with a hyphen (-) and a right angle bracket (>), with no space between them

Operation Results on #MISSING Values and Zero (0) Values

If a data value does not exist for a unique combination of members, Essbase gives the combination a value of #MISSING. A #MISSING value is different from a zero (0) value. Therefore, Essbase treats #MISSING values differently from 0 values.

The following tables shows how Essbase calculates #MISSING values. In this table, X represents any number.

Calculation/Operation	Result
$X + \text{\#MISSING}$	X
$X - \text{\#MISSING}$ $\text{\#MISSING} - X$	X -X
$X * \text{\#MISSING}$	#MISSING
$X / \text{\#MISSING}$ $\text{\#MISSING} / X$ $X / 0$	#MISSING #MISSING #MISSING
$X \% \text{\#MISSING}$ $\text{\#MISSING} \% X$ $X \% 0$	#MISSING #MISSING #MISSING
$X == \text{\#MISSING}$	False, unless X is #MISSING
$X != \text{\#MISSING}$ $X <> \text{\#MISSING}$	True, unless X is #MISSING True, unless X is #MISSING
$(X <= \text{\#MISSING})$	$(X <= 0)$
$(X >= \text{\#MISSING})$	$(X >= 0)$ or $(X == \text{\#MISSING})$
$(X > \text{\#MISSING})$	$(X > 0)$
$(X < \text{\#MISSING})$	$(X < 0)$
X AND #MISSING: 1 AND #MISSING* 0 AND #MISSING #MISSING AND #MISSING	#MISSING 0 #MISSING
X OR #MISSING: 1 OR #MISSING ¹ 0 OR #MISSING #MISSING OR #MISSING	1 #MISSING #MISSING
IF (#MISSING)	IF (0)
f (#MISSING)	#MISSING for any Essbase function of one variable
f (X)	#MISSING for any X not in the domain of f, and any Essbase function of more than one variable (except where specifically noted)

¹1 represents any nonzero value.

For information on how Essbase aggregates #MISSING values, see the *Oracle Essbase Database Administrator's Guide*.

Calculation Function Categories

This section lists all of the Essbase calculation functions, grouped by function type.

- “Conditional and Logical Operators” on page 25
- “Boolean Functions” on page 27
- “Relationship Functions” on page 28
- “Calculation Operators” on page 24
- “Mathematical Functions” on page 29
- “Member Set Functions” on page 30
- “Range and Financial Functions” on page 33
- “Allocation Functions” on page 35
- “Forecasting Functions” on page 35
- “Statistical Functions” on page 36
- “Date & Time Function” on page 36
- “Miscellaneous Functions” on page 37
- “Custom-defined Functions” on page 37

Boolean Functions

A Boolean function returns TRUE or FALSE (1 or 0, respectively). Boolean functions are generally used in conjunction with the IF command to provide a conditional test. Because they generate a numeric value, however, Boolean functions can also be used as part of a member formula.

Boolean functions are useful because they can determine which formula to apply based on characteristics of the current member combination. For example, you may want to restrict a calculation to those members in a dimension that contain input data. In this case, you preface the calculation with an IF test that is based on @ISLEV (*dimName*, 0).

If one of the function parameters is a cross-dimensional member; for example, @@ISMBR (Sales->Budget), all parts of the cross-dimensional member must match all parts of the current cell to return a value of TRUE.

In the following quick-reference table, "the current member" means the member that is currently being calculated by the function. Words in italics, such as *member*, loosely indicate information you supply to the function. For details, see the individual function topics.

Function	Condition Tested
@ISACCTYPE	Whether the current member has a particular <i>accounts tag</i> .
@ISANCEST	Whether the current member is an ancestor of <i>member</i> .
@ISCHILD	Whether the current member is a child of <i>member</i> .
@ISDESC	Whether the current member is a descendant of <i>member</i> .
@ISGEN	Whether the current member of <i>dimension</i> is in <i>generation</i> .
@ISIANCEST	Whether the current member is the same member or an ancestor of <i>member</i> .
@ISICHILD	Whether the current member is the same member or a child of <i>member</i> .
@ISIDESC	Whether the current member is the same member or a descendant of <i>member</i> .
@ISIPARENT	Whether the current member is the same member or the parent of <i>member</i> .
@ISISIBLING	Whether the current member is the same member or a sibling of <i>member</i> .
@ISLEV	Whether the current member of <i>dimension</i> is in <i>level</i> .
@ISMBR	Whether the current member is <i>member</i> , or is found in <i>member list</i> , or is found in a <i>range</i> returned by another function.
@ISPARENT	Whether the current member is the parent of <i>member</i> .
@ISSAMEGEN	Whether the current member is in the same generation as <i>member</i> .
@ISSAMELEV	Whether the current member is in the same level as <i>member</i> .
@ISSIBLING	Whether the current member is a sibling of <i>member</i> .
@ISUDA	Whether the current member of <i>dimension</i> has a particular <i>user-defined attribute string</i> .

Relationship Functions

Relationship functions look up specific values within the database based on current cell location and a series of parameters. You can use these functions to refer to another value in a data series. Relationship functions have an implicit current member argument; that is, these functions are dependent on the current member's position.

In the following quick-reference table, words in italics loosely represent information you supply to the function. For details, see the individual function topics.

Function	Return Value
@ANCESTVAL	Ancestor values of a specified <i>one-dimensional member combination</i> .
@ATTRIBUTEVAL	Associated attribute value from a <i>Boolean attribute dimension</i> .
@ATTRIBUTESVAL	Associated attribute value from a <i>text attribute dimension</i> .

Function	Return Value
@ATTRIBUTEVAL	Associated attribute value from a <i>numeric or date attribute dimension</i> .
@CURGEN	Generation number of the current member in <i>dimension</i> .
@CURLEV	Level number of the current member in <i>dimension</i> .
@GEN	Generation number of <i>member</i> .
@LEV	Level number of <i>member</i> .
@MDANCESTVAL	Ancestor values for any number of <i>multidimensional member combinations</i> .
@MDPARENTVAL	Parent values for any number of <i>multidimensional member combinations</i> .
@PARENTVAL	Parent values for <i>member in dimension</i> .
@SANCESTVAL	Ancestor values for shared members at a certain <i>depth</i> under <i>root member</i> .
@SPARENTVAL	Parent values for shared members under <i>root member</i> .
@XREF	Values from a different database than the one being calculated.
@XWRITE	Writes values to a different database than the one being calculated.

Mathematical Functions

These functions perform specific mathematical calculations. Mathematical functions define and return values that are based on selected member expressions. These functions cover many basic statistical functions and return numeric results that are based on supplied member values. Advanced statistical functions are included in the statistical functions category.

In the following quick-reference table, words in italics loosely represent information you supply to the function. For details, see the individual function topics.

Function	Return Value
@ABS	Absolute value of <i>expression</i> .
@AVG	Average of all values in <i>expList</i> .
@EXP	e (base of natural logarithms) raised to the power of <i>expression</i> .
@FACTORIAL	Factorial of <i>expression</i> .
@INT	Next lowest integer value of <i>expression</i> .
@LN	e (base of natural logarithms) of <i>expression</i> .
@LOG	Any <i>base</i> logarithm of <i>expression</i> .
@LOG10	Base-10 logarithm of <i>expression</i> .

Function	Return Value
@MAX	Maximum value found in cells of <i>expression list</i> .
@MAXS	Maximum value found in cells of <i>expression list</i> , optionally skipping empty values.
@MIN	Minimum value found in cells of <i>expression list</i> .
@MINS	Minimum value found in cells of <i>expression list</i> , optionally skipping empty values.
@MOD	Modulus of a division operation between two <i>members</i> .
@POWER	<i>Expression</i> raised to <i>power</i> .
@REMAINDER	Remainder value of <i>expression</i> .
@ROUND	<i>Expression</i> rounded to <i>numDigits</i> .
@SUM	Sum of values found in cells of <i>expression list</i> .
@TRUNCATE	<i>Expression</i> with fractional part removed, returning an integer.
@VAR	Variance between two <i>members</i> .
@VARPER	Percent variance between two <i>members</i> .

Member Set Functions

Member set functions return a list of members. This list is based on the member specified and the function used. You can use operators to specify [Generation and Level Range Operators for Member Set Functions](#) with member set functions.

When a member set function is called as part of a formula, the list of members is generated before the calculation begins. The list never varies because it is based on the specified member and is independent of the current member.

If a member set function (for example, @CHILDREN or @SIBLINGS) is used to specify the list of members to calculate in a calculation script, Essbase bypasses the calculation of any Dynamic Calc or Dynamic Calc and Store members in the resulting list.

Only the @ATTRIBUTE and @WITHATTR functions can use attribute members or members of the Attribute Calculations dimension as parameters in member set functions.

You can use cross-dimension expressions such as ("1998":"2001" -> @Levmbrr (Year, 0)). The cross-dimensional operator is associative ($x \rightarrow y \rightarrow z = x \rightarrow (y \rightarrow z)$), but not commutative because $x \rightarrow y = y \rightarrow x$ is a set, but the order of elements is different.

Function	Return Value
@ALLANCESTORS	All ancestors of <i>member</i> , including ancestors of shared <i>member</i> .
@ANCEST	Ancestor at <i>distance</i> from the current member or an explicitly specified <i>member</i> .
@ANCESTORS	All ancestors of <i>member</i> , or those ancestors up to a specified <i>distance</i> .

Function	Return Value
@ATTRIBUTE	All base members associated with <i>attribute member name</i> .
@BETWEEN	All members whose name string value fall between, and are inclusive of, two specified string tokens.
@CHILDREN	Children of <i>member</i> .
@CURRMBR	Member currently being calculated in the specified <i>dimension</i> .
@DESCENDANTS	All descendants of <i>member</i> , or those descendants down to a specified <i>distance</i> .
@EQUAL	Member names that match the specified token name.
@EXPAND	Expands a member search by calling a member set function for each member in a member list.
@GENMBRS	Members of <i>dimension</i> that are at <i>generation</i> .
@IALLANCESTORS	<i>Member</i> and ancestors of <i>member</i> , including ancestors of shared <i>member</i> .
@IANCESTORS	<i>Member</i> , and either all member ancestors or those ancestors up to a specified <i>distance</i> .
@ICHILDREN	<i>Member</i> and its children.
@IDESCENDANTS	<i>Member</i> , and either all member descendants or those descendants down to a specified <i>distance</i> .
@ILANCESTORS	Members of the specified list of members, and either all ancestors of the specified list of members or those ancestors up to a specified <i>distance</i> .
@ILDESCENDANTS	Members of the specified list of members, and either all descendants of the specified list of members or those descendants down to a specified <i>distance</i> .
@ILSIBLINGS	<i>Member</i> and its left siblings.
@IRSIBLINGS	<i>Member</i> and its right siblings.
@IRDESCENDANTS	<i>Member</i> and all its descendants, or those descendants down to a specified <i>distance</i> , including descendants of shared <i>member</i> .
@ISIBLINGS	<i>Member</i> and its siblings.
@LANCESTORS	All ancestors of the specified list of <i>members</i> , or those ancestors up to a specified <i>distance</i> .
@LDESCENDANTS	All descendants of the specified list of <i>members</i> , or those descendants down to a specified <i>distance</i> .
@LEVMBRS	Members of <i>dimension</i> that are at <i>level</i> .
@LIST	A single list compiled from <i>arguments</i> , and can be used for functions requiring an expression list, a member list, or a range list.
@LSIBLINGS	Left siblings of <i>member</i> .
@MATCH	Members that match a <i>pattern</i> search performed over a <i>generation</i> , a <i>level</i> , or a <i>member</i> and its descendants.
@MBRCOMPARE	Member names that match the comparison criteria.
@MBRPARENT	Parent of the specified member.

Function	Return Value
@MEMBER	Member with name <i>string</i> .
@MERGE	Merged list from two <i>lists</i> .
@NEXTSIBLING	Next, or right-most, sibling of <i>member</i> .
@NOTEQUAL	Member names that do not match the specified token name.
@PARENT	Parent of the current member being calculated in <i>dimension</i> , optionally crossed with another <i>member</i> .
@PREVSIBLING	Previous, or left-most, sibling of <i>member</i> .
@RANGE	Member list that crosses a <i>member</i> from one dimension with a <i>range</i> from another dimension.
@RDESCENDANTS	All descendants of <i>member</i> , or those down to a specified <i>distance</i> , including descendants of shared <i>member</i> .
@RELATIVE	All members that are at <i>distance</i> from <i>member</i> .
@REMOVE	<i>List1</i> , with anything that is also in <i>list2</i> removed.
@RSIBLINGS	Right siblings of <i>member</i> .
@SHIFTSIBLING	Sibling at specified <i>distance</i> from <i>member</i> .
@SIBLINGS	Siblings of <i>member</i> .
@UDA	Members of <i>dimension</i> that have <i>UDA</i> .
@WITHATTR	Base members from <i>dimension</i> that are associated with an attribute meeting a <i>condition</i> .
@XRANGE	Range of members between (and inclusive of) two <i>members</i> at the same level.

Generation and Level Range Operators for Member Set Functions

The operators `:` and `::` can be used with member set functions, which return a list of members. The `:` operator returns level-based ranges and the `::` operator returns generation-based ranges. For example, `Jan:Dec` and `Jan::Dec` both return all members between and inclusive of Jan and Dec.

The difference is that `Jan:Dec` returns all members at the same level and `Jan::Dec` returns all members at the same generation.

For example, if we have the outline:

```

Q1 - Jan
    Feb
    Mar
Q2 - Apr
    May
    Jun
Q3
Q4 - Oct
    Nov
    Dec

```


The function @MOVAVG(Sales, 3, Jan:Dec) computes @MOVAVG(Sales, 3, Jan, Feb, Mar, Apr, May, Jun, Q3, Oct, Nov, Dec).

The function @MOVAVG(Sales, 3, Jan::Dec) computes @MOVAVG(Sales, 3, Jan, Feb, Mar, Apr, May, Jun, Oct, Nov, Dec).

Range and Financial Functions

Range functions take a range of members as an argument. Rather than return a single value, these functions calculate a series of values internally based on the range specified.

Financial functions execute specialized financial calculations.

Function	Return Value
@ACCUM	The sum of values of a specified member across a range
@AVGRANGE	The average of values of a specified member across a range
@COMPOUND	The compound interest of values of a specified member across a range, calculated at a specified rate
@COMPOUNDGROWTH	A series of values that represent the compound growth of the specified member across a range of members, calculated at a specified rate
@CURRMBRRANGE	A range of members that is based on the relative position of the member combination Essbase is currently calculating
@DECLINE	Depreciation of a member over a specified period, calculated using the declining balance method
@DISCOUNT	Discounted values of a specified member, calculated at a specified rate, across a range of values from the time dimension
@GROWTH	A series of values that represents the linear growth of the specified value
@INTEREST	A series of values that represent the linear growth of a specified member, calculated at a specified rate, across a range of members from the time dimension
@IRR	The internal rate of return on a cash flow calculated across the time dimension or a specified range of members
@MAXRANGE	The maximum value of a member across a range of members
@MAXSRANGE	The maximum value of a member across a range of members, with the ability to skip zero and #MISSING values
@MDSHIFT	The next or <i>n</i> th member in a range of members, retaining all other members identical to the current member across multiple dimensions
@MINRANGE	The minimum value of a member across a range of members
@MINSRANGE	The minimum value of a member across a range of members, with the ability to skip zero and #MISSING values
@NEXT	The next or <i>n</i> th member in a range of members
@NEXTS	The next or <i>n</i> th member in a range of members, with the option to skip #MISSING, zero, or both values
@NPV	The Net Present Value of an investment based on a series of payments and income values

Function	Return Value
@PTD	The period-to-date values of members in the time dimension
@PRIOR	A list of the previous or <i>n</i> th previous members in a range of members
@PRIORS	A list of the previous or <i>n</i> th previous members in a range of members, with the option to skip #MISSING, zero, or both values
@RANGE	A member list that crosses the specified member from one dimension with the specified member range from another dimension
@SHIFT @SHIFTPLUS @SHIFTMINUS	A list of the next or <i>n</i> th members in a range of members, retaining all other members identical to the current member and in the specified dimension
@SLN	Depreciation amounts, across a range period, that an asset in the current period may be depreciated, calculated using the straight-line depreciation method
@SUMRANGE	A list of summarized values of all specified members across a range of members
@SYD	Depreciation amounts, across a range of periods, of an asset in the current period, calculated using the sum of the year's digits depreciation method
@XRANGE	A list of a range of members between specified members at the same level

Range List Parameters

Some range and forecasting functions recognize the optional parameter *rangeList* or *XrangeList* as the last parameter. *rangeList* is a range of members from one dimension; *XrangeList* is a range of members from one or more dimensions.

If *rangeList* or *XrangeList* is not given, the level 0 (leaf) members from the dimension tagged as Time become the default range. If no dimension is tagged as Time and the last parameter is not given, Essbase reports a syntax error.

The following table provides examples of valid values for *rangeList* or *XrangeList*.

Example	Description
Mar99	A single member
Mar99, Apr99, May99	A comma-delimited list of members.
Jan99:Dec99	A level range. A level range includes all members on the same level between and including the members defining the range.
Q1_99::Q4_2000	A generation range. A generation range includes the members defining the range and all members that are within the range and of the same generation.
Q1_99::Q4_2000, FY98, FY99, FY2000	A generation range and a comma-delimited list

Example	Description
@SIBLINGS(Dept01), Dept65:Dept73, Total_Dept	A member set function and one or more range lists

The following table provides examples of valid values for *XrangeList*.

Example	Description
Jan->Actual->Sales, Dec->Actual->Sales	A comma-delimited list of members from one or more dimensions.
Actual->Jan, @XRANGE(Actual->December, Budget->Mar);	A comma-delimited list and a range.
@XRANGE(Jan->Actual,Dec->Budget);	A @XRANGE function.
@CHILDREN("Colas"),@CHILDREN("West")	A member set function as part of a range list.

Financial functions never return a value; rather, they internally calculate a series of values based on the range specified and write the results to a range of cells. Thus, you cannot apply any operator directly to the function.

Allocation Functions

These functions allocate values that are input at the parent level. The values are allocated across child members in one or more dimensions, based on specified criteria. These functions consolidate the common tasks that are required to perform allocations in Essbase.

Function	Allocation Type
@ALLOCATE	Allocates values to lower-level members in one level.
@MDALLOCATE	Allocates values to lower-level members in multiple dimensions.

Forecasting Functions

Forecasting functions manipulate data for the purpose of smoothing, interpolating, or calculating future values. Forecasting functions are often used in planning, analysis, and modeling applications. Some forecasting functions recognize the optional [Range List Parameters](#) (rangeList or XrangeList).

Function	Data Manipulation
@MOVAVG	Applies a moving average to a data set, replacing each term in the list with a trailing average. This function modifies the data set for smoothing purposes.
@MOVMAX	Applies a moving maximum to a data set, replacing each term in the list with a trailing maximum. This function modifies the data set for smoothing purposes.

Function	Data Manipulation
@MOV MED	Applies a moving median to a data set, replacing each term in the list with a trailing median. This function modifies the data set for smoothing purposes.
@MOV MIN	Applies a moving minimum to a data set, replacing each term in the list with a trailing minimum. This function modifies the data set for smoothing purposes.
@MOV SUM	Applies a moving sum to a data set. This function modifies the data set for smoothing purposes.
@MOV SUM X	Applies a moving sum to a data set, enabling specification of values for trailing members. This function modifies the data set for smoothing purposes.
@SPLINE	Applies a smoothing spline to a set of data points. A spline is a mathematical curve that is used to smooth or interpolate data.
@TREND	Calculates future values, basing the calculation on curve-fitting to historical values

Statistical Functions

Statistical functions calculate advanced statistical values, such as correlation or variance. These functions are often used in sales and marketing applications.

Function	Return Value
@CORRELATION	The correlation coefficient between two parallel data sets
@COUNT	The number of data values in the specified data set
@MEDIAN	The median (middle value) of the specified data set
@MODE	The mode (the most frequently occurring value) in the specified data set
@RANK	The rank (position in the sorted data set) of the specified members or the specified value among the values in the specified data set.
@STDEV	The standard deviation of the specified data set
@STDEVP	The standard deviation of the specified data set, calculated over the entire population
@STDEV RANGE	The standard deviation of all values of the specified member across the specified data set. The specified mbrName is crossed with a range list to obtain the sample across which the standard deviation is calculated.
@VARIANCE	The statistical variance of the specified data set (expList), based upon a sample of a population
@VARIANCE P	The statistical variance of the specified data set (expList), based upon the entire population

Date & Time Function

The date function, @TODATE, converts date strings to numbers that can be used in calculation formulas.

Miscellaneous Functions

- [@CALCMODE](#)—This function enables you to specify whether a formula is calculated in cell mode or block mode and whether a formula is calculated bottom-up or top-down
- [@CONCATENATE](#), [@SUBSTRING](#), and [@NAME](#)—These functions enable manipulation of character strings.
- [@RETURN](#)—This function enables termination of a calculation, with a custom error message.

Custom-defined Functions

This custom-defined group is a category of functions that you develop for calculation operations that are not enabled by the built-in Essbase functions. Custom-defined functions are written in the Java programming language and registered on the server. The Essbase calculator framework calls custom-defined functions as external functions. For more details, see [Create Macro](#) and [Create Function](#) in MaxL.

Calculation Function Reference

Consult the Contents pane for a categorical list of calculation functions.

@ABS	@ISANCEST	@MOVSUMX
@ACCUM	@ISATTRIBUTE	@NAME
@ALLANCESTORS	@ISCHILD	@NEXT
@ALIAS	@ISDESC	@NEXTS
@ALLOCATE	@ISGEN	@NEXTSIBLING
@ANCEST	@ISIANCEST	@NOTEQUAL
@ANCESTORS	@ISISIBLINGS	@NPV
@ANCESTVAL	@ISICHILD	@PARENT
@ATTRIBUTE	@ISIDESC	@PARENTVAL
@ATTRIBUTEVAL	@ISIPARENT	@POWER
@ATTRIBUTESVAL	@ISISIBLING	@PREVSIBLING
@ATTRIBUTEVAL	@ISLEV	@PRIOR
@AVG	@ISMBR	@PRIORS
@AVGRANGE	@ISMBRWITHATTR	@PTD
@BETWEEN	@ISPARENT	@RANGE
@CALCMODE	@ISSAMEGEN	@RANK

@ABS	@ISANCEST	@MOVSUMX
@CHILDREN	@ISSAMELEV	@RDESCENDANTS
@COMPOUND	@ISSIBLING	@RELATIVE
@COMPOUNDGROWTH	@ISUDA	@REMAINDER
@CONCATENATE	@LANCESTORS	@REMOVE
@CORRELATION	@LDESCENDANTS	@RETURN
@COUNT	@LEV	@ROUND
@CURGEN	@LEVMBRS	@RSIBLINGS
@CURLEV	@LIKE	@SANCESTVAL
@CURRMBR	@LIST	@SHARE
@CURRMBRRANGE	@LN	@SHIFT
@DATEDIFF	@LOG	@SHIFTMINUS
@DATEPART	@LOG10	@SHIFTPLUS
@DATEROLL	@LSIBLINGS	@SHIFTSIBLING
@DECLINE	@MATCH	@SIBLINGS
@DESCENDANTS	@MAX	@SLN
@DISCOUNT	@MAXRANGE	@SPARENTVAL
@ENUMVALUE	@MAXS	@SPLINE
@EQUAL	@MAXSRANGE	@STDEV
@EXP	@MBRCOMPARE	@STDEVP
@EXPAND	@MBRPARENT	@STDEV RANGE
@FACTORIAL	@MDALLOCATE	@SUBSTRING
@FORMATDATE	@MDANCESTVAL	@SUM
@GEN	@MDPARENTVAL	@SUMRANGE
@GENMBRS	@MDSHIFT	@SYD
@GROWTH	@MEDIAN	@TODATE
@IALLANCESTORS	@MEMBER	@TODATEEX
@IANCESTORS	@MERGE	@TODAY
@ICHILDREN	@MIN	@TREND
@IDESCENDANTS	@MINRANGE	@TRUNCATE

@ABS	@ISANCEST	@MOVSUMX
@ILANCESTORS	@MINS	@UDA
@ILDESCENDANTS	@MINSRANGE	@VAR
@ILSIBLINGS	@MOD	@VARPER
@INT	@MODE	@VARIANCE
@INTEREST	@MOVAVG	@VARIANCEP
@IRDESCENDANTS	@MOVMAX	@WITHATTR
@IRR	@MOVMED	@XRANGE
@IRSIBLINGS	@MOVMIN	@XREF
@ISACCTYPE	@MOVSUM	@XWRITE

@ABS

Returns the absolute value of *expression*. The absolute value of a number is that number less its sign. A negative number becomes positive, while a positive number remains positive.

Syntax

@ABS (*expression*)

Parameter Description

expression Member name or mathematical expression that generates a numeric value.

Example

The following example is based on the Demo Basic database. In this example, Variance needs to be presented as a positive number. The @ABS function is used because otherwise some combinations of Actual - Budget would return negative values.

```
Variance=@ABS(Actual-Budget);
```

This example produces the following report:

Sales	VCR	San_Francisco	
	Jan	Feb	Mar
	===	===	===
Actual	1,323	1,290	1,234
Budget	1,200	1,100	1,100
Variance	123	190	134

See Also

- [@INT](#)
- [@REMAINDER](#)
- [@ROUND](#)
- [@TRUNCATE](#)

@ACCUM

Accumulates the values of *mbrName* within *rangeList*, up to the current member in the dimension of which *rangeList* is a part.

Syntax

```
@ACCUM (mbrName [, rangeList])
```

Parameter Description

mbrName Any valid single member name or member combination (or a function that returns a single member or member combination) whose value is to be accumulated.

rangeList Optional comma-delimited list of members, member set functions, or range functions, across which the accumulation occurs. If *rangeList* is not specified, Essbase uses the level 0 members from the dimension tagged as Time.

Notes

- Financial functions never return a value; rather, they calculate a series of values internally based on the range specified.
- @ACCUM accepts the @ATTRIBUTE member set function as a member range.
- If you use an Essbase member set function to generate a member list for the *rangeList* parameter (for example, @SIBLINGS), to ensure correct results, consider the order in which Essbase sorts the generated member list. For more information, see the *Oracle Essbase Technical Reference* topic for the member set function you are using.
- You cannot apply an operator (for example divide or multiply) to @Accum. For example, the formula Budget=@ACCUM(Actual, Jan:Feb)/2 is not valid.

Example

In this example, Accum Asset is calculated using the following formula:

```
"Accum Asset" = @ACCUM(Asset, FY1997:FY2002);
```

This example produces the following report. This report shows that the values for Asset are accumulated starting with FY1997 and the yearly accumulation value is placed in Accum Asset for FY1997 through FY2002:

	FY1997	FY1998	FY1999	FY2000	FY2001	FY2002
	=====	=====	=====	=====	=====	=====
Asset	9,000	0	1,000	0	2,500	1,500
Residual	750	0	0	0	#MISSING	#MISSING
Life	5	0	3	0	#MISSING	#MISSING
Accum Asset	#MISSING	#MISSING	1,000	1,000	3,500	5,000

The value of Accum Asset is #MISSING for FY1997 because that is the starting year. The value of Accum Asset is #MISSING for FY1998 because there was no accumulation that year. For FY1999, the value of the asset grew by 1,000, so Accum Asset has a value of 1000.

@ALLANCESTORS

Returns all ancestors of the specified member, including ancestors of any occurrences of the specified member as a shared member. This function excludes the specified member.

Syntax

```
@ALLANCESTORS (mbrName)
```

Parameter Description

mbrName Any valid single member name or member combination, or a function that returns a single member or member combination.

Notes

- Essbase sorts the generated list of members in ascending order of the member number in the outline. Using Sample Basic as an example, if you specify 100-20 for *mbrName*, 100, Diet, and Product are returned (in that order). However, the order in which shared ancestors are returned is not guaranteed. This order is important to consider when you use the @ALLANCESTORS member set function with certain forecasting and statistical functions.
- You can use @ALLANCESTORS as a parameter of another function, where that parameter is a list of members.

Example

The following example is based on the Sample Basic database. Sample Basic has a shared level of diet drinks, which includes 100-20 (Diet Cola). So 100-20 (Diet Cola) is a descendant of 100 (Colas) and is a shared member descendant of Diet:

```
100
    100-10
    100-20
    ...
Diet
    100-20 (Shared Member)
    ...
```

The following calculation script increases by 5% the Budget->Sales values of all ancestors of 100-20, including Diet.

```
FIX(Budget, @ALLANCESTORS("100-20"))
Sales = Sales * 1.05;
ENDFIX
```

This example produces the following report. This report shows that the Budget->Sales values for 100, Diet, and Product (the ancestors of 100-20) have been increased by 5%. The original values were 8980, 8260, and 28480, respectively.

		Jan	
		Actual	Budget
		Sales	Sales
		=====	=====
Market	100-10	4860	5200
	100-20	2372	2610
	100-30	1082	1170

100	8314	9429	*
100-20	2372	2610	
200-20	3122	3090	
300-30	2960	2560	
Diet	8454	8673	*
Product	31538	30954	*

See Also

- [@IALLANCESTORS](#)
- [@LANCESTORS](#)
- [@ILANCESTORS](#)

@ALIAS

Takes a string as an argument and returns an alias name to the function that calls @ALIAS.

Syntax

```
@ALIAS (function_name)
```

Notes

Because functions that take strings as arguments may not function correctly if the string matches a member alias, use the function @ALIAS to pass member alias names as strings, for example when passing alias names as strings to functions such as @ISUDA, @UDA, @CONCATENATE, @SUBSTRING, @MATCH, or @NAME.

Example

For example, if the value "US\$" is both an alias and a user-defined attribute, pass the string using @ALIAS:

```
IF (@ISUDA (@ALIAS ("US$")))
...
ENDIF
```

@ALLOCATE

Allocates values from a member, from a cross-dimensional member, or from a value across a member list. The allocation is based on a variety of criteria.

This function allocates values that are input at an upper level to lower-level members. The allocation is based upon a specified share or spread of another variable. For example, you can allocate values loaded to a parent member to all of that member's children. You can specify a rounding parameter for allocated values and account for rounding errors.

Syntax

```
@ALLOCATE (amount, allocationRange, basisMbr, [roundMbr], method [, methodParams] [, round [, numDigits] [, roundErr]])
```

Parameter	Description
amount	<p>A value, member, or cross-dimensional member that contains the value to be allocated into <i>allocationRange</i>. The value may also be a constant.</p> <ul style="list-style-type: none"> ● If <i>amount</i> is a member, the member must be from the dimension to which <i>allocationRange</i> belongs. ● If <i>amount</i> is a cross-dimensional member, at least one of its members must be from the dimension to which <i>allocationRange</i> belongs. ● If no member or cross-dimensional member is from the dimension to which <i>allocationRange</i> belongs, a warning message is displayed. <p>If the <i>amount</i> parameter is a loaded value, it cannot be a Dynamic Calc member.</p>
allocationRange	<p>A comma-delimited list of members, member set functions, or range functions, into which value(s) from <i>amount</i> are allocated. <i>allocationRange</i> should be from only one level (for example, @CHILDREN(Total Expenses) rather than from multiple levels (for example, @DESCENDANTS(Product))).</p>
basisMbr	<p>A value, member, or cross-dimensional member that contains the values that provide the basis for the allocation. The <i>method</i> you specify determines how the basis data is used.</p>
roundMbr	<p>Optional. The member or cross-dimensional member to which rounding errors are added. The member (or at least one member of a cross-dimensional member) must be included in <i>allocationRange</i>.</p>

Parameter	Description
method	<p>The expression that determines how values are allocated. One of the following:</p> <ul style="list-style-type: none"> <p>share:</p> <p>Uses <i>basisMbr</i> to calculate a percentage share. The percentage share is calculated by dividing the value in <i>basisMbr</i> for the current member in <i>allocationRange</i> by the sum across the <i>allocationRange</i> for that basis member:</p> $\text{amount} * (@CURRMBR()->\text{basisMbr}/@SUM(\text{allocationRange}->\text{basisMbr}))$ <p>spread:</p> <p>Spreads <i>amount</i> across <i>allocationRange</i>:</p> $\text{amount} * (1/@COUNT(SKIP, \text{allocationRange}))$ <p>SKIPNONE SKIPMISSING SKIPZERO SKIPBOTH: Values to be ignored during calculation of the spread. You must specify a SKIP parameter only for <i>spread</i>.</p> <ul style="list-style-type: none"> SKIPNONE: Includes all cells. SKIPMISSING: Excludes all #MISSING values in <i>basisMbr</i>, and stores #MISSING for values in <i>allocationRange</i> for which the <i>basisMbr</i> is missing. SKIPZERO: Excludes all zero (0) values in <i>basisMbr</i>, and stores #MISSING for values in <i>allocationRange</i> for which the <i>basisMbr</i> is zero. SKIPBOTH: Excludes all zero (0) values and all #MISSING values, and stores #MISSING for values in <i>allocationRange</i> for which the <i>basisMbr</i> is zero (0) or #MISSING. <p>percent: Takes a percentage value from <i>basisMbr</i> for each member in <i>allocationRange</i> and applies the percentage value to <i>amount</i>:</p> $\text{amount} * (@CURRMBR()->\text{basisMbr} * .01)$ <p>add: Takes the value from <i>basisMbr</i> for each member of <i>allocationRange</i> and adds the value to <i>amount</i>:</p> $\text{amount} + @CURRMBR()->\text{basisMbr}$ <p>subtract: Takes the value from <i>basisMbr</i> for each member of <i>allocationRange</i> and subtracts the value from <i>amount</i>:</p> $\text{amount} - @CURRMBR()->\text{basisMbr}$ <p>multiply: Takes the value from <i>basisMbr</i> for each member of <i>allocationRange</i> and multiplies the value by <i>amount</i>:</p> $\text{amount} * @CURRMBR()->\text{basisMbr}$ <p>divide: Takes the value from <i>basisMbr</i> for each member of <i>allocationRange</i> and divides the value by <i>amount</i>:</p> $\text{amount}/@CURRMBR()->\text{basisMbr}$
round	<p>Optional. One of the following:</p> <ul style="list-style-type: none"> noRound: No rounding. <i>noRound</i> is the default. roundAmt: Indicates that you want to round the allocated values. If you specify <i>roundAmt</i>, you also must specify <i>numDigits</i> to indicate the number of decimal places to round to.

Parameter	Description
numDigits	<p>An integer that represents the number of decimal places to round to. You must specify <i>numDigits</i> if you specify <i>roundAmt</i>.</p> <ul style="list-style-type: none"> ● If <i>numDigits</i> is 0, the allocated values are rounded to the nearest integer. The default value for <i>numDigits</i> is 0. ● If <i>numDigits</i> is greater than 0, the allocated values are rounded to the specified number of decimal places. ● If <i>numDigits</i> is a negative value, the allocated values are rounded to a power of 10. <p>If you specify <i>roundAmt</i>, you also can specify a <i>roundErr</i> parameter.</p>
roundErr	<p>Optional. An expression that specifies where rounding errors should be placed. You must specify <i>roundAmt</i> in order to specify <i>roundErr</i>. If you do not specify <i>roundErr</i>, rounding errors are discarded.</p> <p>To specify <i>roundErr</i>, choose from one of the following:</p> <ul style="list-style-type: none"> ● <i>errorsToHigh</i>: Adds rounding errors to the member with the highest allocated value. If allocated values are identical, adds rounding errors to the first value in <i>allocationRange</i>. (For this option, Essbase does not distinguish between #MI and zero values.) ● <i>errorsToLow</i>: Adds rounding errors to the member with the lowest allocated value. If allocated values are identical, adds rounding errors to the first value in <i>allocationRange</i>. #MISSING is treated as the lowest value in a list; if multiple values are #MISSING, rounding errors are added to the first #MISSING value in the list. ● <i>errorsToMbr</i>: Adds rounding errors to the specified <i>roundMbr</i>, which must be included in <i>allocationRange</i>.

Notes

- When you use @ALLOCATE in a calculation script, use it within a FIX statement; for example, FIX on the member to which the allocation amount is loaded. Although FIX is not required, using it may improve calculation performance.
- If you use @ALLOCATE in a member formula, your formula should look like this:

```
Member Name = @ALLOCATE (...)
```

This is because allocation functions never return a value; rather, they calculate a series of values internally based on the range specified.
- For an example that explains the use of rounding error processing with the @ALLOCATE function, see the *Oracle Essbase Database Administrator's Guide*.

Example

Consider the following example from the Sample Basic database. The example assumes that the Scenario dimension contains an additional member, PY Actual, for the prior year's actual expenses. Data values of 7000 and 8000 are loaded into Budget->Total Expenses for Jan and Feb, respectively. (For this example, assume that Total Expenses is not a Dynamic Calc member.)

You need to allocate values to each expense category (to each child of Total Expenses). The allocation for each of child of Total Expenses is based on the child's share of actual expenses for the prior year (PY Actual):

```
FIX("Total Expenses")
Budget = @ALLOCATE(Budget->"Total Expenses",@CHILDREN("Total Expenses"),
```

```
"PY Actual", , share);
ENDFIX
```

This example produces the following report:

	Product		Market	
	PY Actual		Budget	
	Jan	Feb	Jan	Feb
	===	===	===	===
Marketing	5223	5289	3908.60	4493.63
Payroll	4056	4056	3035.28	3446.05
Misc	75	71	56.13	60.32
Total Expenses	9354	9416	7000	8000

See Also

- [@MDALLOCATE](#)

@ANCEST

Returns the ancestor at the specified generation or level of the current member being calculated in the specified dimension. If you specify the optional *mbrName*, that ancestor is combined with the specified member.

This member set function can be used as a parameter of another function, where that parameter is a member or list of members.

Syntax

```
@ANCEST (dimName, genLevNum [, mbrName])
```

Parameter Description

dimName	Single dimension name specification.
genLevNum	An integer value that defines the generation or level number from which the ancestor value is returned. A positive integer defines a generation number. A value of 0 or a negative integer defines a level number.
mbrName	Optional. Any valid single member name or member combination, or a function that returns a single member or member combination, that is crossed with the ancestor returned.

Notes

- You cannot use the @ANCEST function in a FIX statement.
- You can use the @ANCEST function on both the left-hand and right-hand sides of a formula. If you use this function on the left-hand side of a formula in a calculation script, associate it with a member. For example:

```
Sales(@ANCEST(Product) = 5);
```

- In some cases, the @ANCEST function is equivalent to the @ANCESTVAL function, except in terms of calculation performance. For example, the following two formulas are equivalent:

```
Sales = @ANCEST(Product, 2);
```

```
Sales = @ANCESTVAL(Product, 2);
```

In this case, using the latter formula results in better calculation performance. In general, use @ANCEST as a member rather than as an implied value of a cell. For example:

```
Sales = @AVG(SKIPMISSING, @SIBLINGS(@ANCEST(Product,2)));
```

- The time required for retrieval and calculation may be significantly longer if this function is in a formula attached to a member tagged as Dynamic Calc or Dynamic Calc and Store.

Example

In the Sample Basic database:

Function	Generated List
@ANCEST(Product,2,Sales)	Colas->Sales, if the current member of Product being calculated is Diet Cola.
@ANCEST(Measures,3,East)	Total Expenses->East, if the current member of Measures being calculated is Payroll.

See Also

- [@PARENT](#)
- [@CHILDREN](#)
- [@ANCESTORS](#)
- [@DESCENDANTS](#)
- [@SIBLINGS](#)

@ANCESTORS

Returns all ancestors of the specified member (*mbrName*) or those up to a specified generation or level. You can use this member set function as a parameter of another function, where that parameter is a list of members.

Syntax

```
@ANCESTORS (mbrName [, genLevNum | genLevName])
```

Parameter	Description
-----------	-------------

<i>mbrName</i>	Any valid single member name or member combination (or a function that returns a single member or member combination).
----------------	------------------------------------------------------------------------------------------------------------------------

<i>genLevNum</i>	Optional. An integer value that defines the absolute generation or level number up to which to select the members. A positive integer defines a generation number. A value of 0 or a negative integer defines a level number.
------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<i>genLevName</i>	Optional. Level name or generation name up to which to select the members.
-------------------	----------------------------------------------------------------------------

Notes

- The generated list of members is sorted starting with the nearest ancestor of the member, followed by the next nearest ancestor of the member, and so on. Using Sample Basic as an example, if you specify @ANCESTORS (200-30), Essbase returns 200, Product (in that order).

This order is important to consider when you use the @ANCESTORS member set function with certain forecasting and statistical functions.

Example

In the Sample Basic database:

```
@ANCESTORS ("New York")
```

returns East, Market (in that order).

```
@ANCESTORS (Qtr4)
```

returns Year.

```
@ANCESTORS ("100-10", 1)
```

returns 100, Product (in that order).

```
@ANCESTORS (Sales, -2)
```

returns Margin, Profit (in that order).

See Also

- [@IANCESTORS](#)
- [@LANCESTORS](#)
- [@ILANCESTORS](#)
- [@ISANCEST](#)
- [@CHILDREN](#)
- [@DESCENDANTS](#)
- [@SIBLINGS](#)

@ANCESTVAL

Returns the ancestor values of a specified member combination.

Syntax

```
@ANCESTVAL (dimName, genLevNum [, mbrName])
```

Parameter	Description
-----------	-------------

<i>dimName</i>	A single dimension name that defines the focus dimension of ancestor values.
----------------	------------------------------------------------------------------------------

<i>genLevNum</i>	Integer value that defines the generation or level number from which the ancestor values are to be returned. A positive integer defines a generation reference. A negative number or value of 0 defines a level reference.
------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<i>mbrName</i>	Optional. Any valid single member name or member combination (or a function that returns a single member or member combination).
----------------	----------------------------------------------------------------------------------------------------------------------------------

Example

In this example, SKU Share is derived by taking Sales in each SKU as a percentage of its product family. Families are at generation 2; therefore, each descendant of family is calculated as a

percentage its respective ancestor. Consolidated results must be calculated for Sales by Product before the SKU Share calculation occurs.

```
"SKU Share" = Sales % @ANCESTVAL(Product,2,Sales);
```

This example produces the following report:

	Sales	SKU Share
	=====	=====
SKU101	510	26.0
SKU102	520	26.5
Group01	1030	52.5
SKU120	430	21.9
SKU123	500	25.5
Group02	930	47.4
Family1	1960	100.00

See Also

- [@MDANCESTVAL](#)
- [@SANCESTVAL](#)
- [@PARENTVAL](#)

@ATTRIBUTE

Lists all base members that are associated with the specified attribute member (*attmbrName*). This member set function can be used as a parameter of another function, where that parameter is a member or list of members.

Syntax

```
@ATTRIBUTE (attMbrName)
```

Parameter Description

attMbrName Single attribute member name.

Notes

When @ATTRIBUTE is used with a non-level 0 member of an attribute dimension, Essbase returns all base members that are associated with the children of the attribute member. For example, in the Sample Basic database, @ATTRIBUTE(Large) returns all base members that fall into one of the population ranges for the attribute parent Large.

If you specify the name of a Boolean attribute dimension (for example, Caffeinated), Essbase returns all base members that are associated with either Caffeinated member (for example, True or False). To return only one, specify the member name (for example, @ATTRIBUTE(Caffeinated_True)).

You may have duplicate Boolean, date, and numeric attribute member names in your outline. For example, 12 can be the attribute value for the size (in ounces) of a product as well as the value for the number of packing units for a product. To distinguish duplicate member names, specify the full attribute member name (for example, @ATTRIBUTE(Ounces_12)).

The generated list of members is sorted in ascending order from the database outline. This order is important to consider when you use the @ATTRIBUTE member set function with certain forecasting and statistical functions.

Example

In the Sample Basic database,

```
@ATTRIBUTE (Can) ;
```

returns all base members with the Can attribute: Cola, Diet Cola, and Diet Cream.

Consider the following two calculation scripts, which are based on the Sample Basic database:

```
/* To increase the marketing budget for markets with large populations */  
FIX (@ATTRIBUTE(Large))  
Marketing = Marketing * 1.1;  
ENDFIX
```

```
/* To calculate the average sales of bottled products */  
"Bottle Sales" = @AVG(SKIPBOTH,@ATTRIBUTE(Bottle));
```

See Also

- [@WITHATTR](#)
- [@ATTRIBUTEVAL](#)

@ATTRIBUTEVAL

Returns, for the current member being calculated, the associated attribute value from the specified Boolean attribute dimension.

Syntax

```
@ATTRIBUTEVAL (attDimName)
```

Parameter	Description
-----------	-------------

attDimName	The name of a Boolean attribute dimension.
------------	--------------------------------------------

Notes

- The @ATTRIBUTEVAL function works only with Boolean attribute dimensions. To return values from numeric or date attribute dimensions, use this [@ATTRIBUTEVAL](#) function. To return values from text attribute dimensions, use the [@ATTRIBUTESVAL](#) function. For more information on types of attributes, see the *Oracle Essbase Database Administrator's Guide*.
- If no attribute is associated with the member being calculated or if the attribute associated with the member is a text, numeric, or date attribute, @ATTRIBUTEVAL returns #MISSING.
- Only level 0 members of attribute dimensions can be associated as attributes of members of a base dimension.

Example

This example is based on the Sample Basic database.

The Product dimension is associated with the Caffeinated Boolean attribute dimension, as shown in the following example:

```
Product {Caffeinated}
  100
    100-10 {Caffeinated:True}
    100-20 {Caffeinated:True}
    100-30 {Caffeinated:False}
  200
    200-10 {Caffeinated:True}
    200-20 {Caffeinated:True}
    200-30 {Caffeinated:False}
    200-40 {Caffeinated:False}
Caffeinated Attribute {Type: Boolean}
  True
  False
```

For the current member of the base dimension Product, the function @ATTRIBUTEVAL(Caffeinated) returns the associated attribute value from the Boolean attribute dimension, Caffeinated. The following table shows the value that would be returned.

Current Member	Return Value
100-10	True
100-20	True
100-30	False
100	#MISSING
200-10	True
200-20	True
200-30	False
200-40	False
200	#MISSING
Product	#MISSING

For any any member that does not have an associated attribute, #MISSING is returned. Only one value is returned at a time.

See Also

- [@ATTRIBUTEVAL](#)
- [@ATTRIBUTESVAL](#)

@ATTRIBUTESVAL

Returns, for the current member being calculated, the associated attribute value from the specified text attribute dimension.

Syntax

@ATTRIBUTESVAL (*attDimName*)

Parameter	Description
-----------	-------------

attDimName	The name of a text attribute dimension.
------------	-----------------------------------------

Notes

- The @ATTRIBUTESVAL function works only with text attribute dimensions. To return values from numeric or date attribute dimensions, use the @ATTRIBUTEVAL function. To return values from Boolean attribute dimensions, use the @ATTRIBUTEVAL function. For more information on types of attributes, see the *Oracle Essbase Database Administrator's Guide*.
- If no attribute is associated with the member being calculated or if the attribute associated with the member is a numeric, Boolean, or date attribute, @ATTRIBUTESVAL returns an empty string.
- Only level 0 members of attribute dimensions can be associated as attributes of members of a base dimension.

Example

This example is based on the Sample Basic database.

The Product dimension is associated with the Pkg Type text attribute dimension, as shown in the following example:

```
Product {Pkg Type}
  100
    100-10 {Pkg Type:Can}
    100-20 {Pkg Type:Can}
    100-30 {Pkg Type:Bottle}
  200
    200-10 {Pkg Type:Bottle}
    200-20 {Pkg Type:Bottle}
    200-30 {Pkg Type:Bottle}
    200-40 {Pkg Type:Bottle}
Pkg Type Attribute {Type: Text}
  Bottle
  Can
```

For the current member of the base dimension, Product, the function

@ATTRIBUTESVAL("Pkg Type")

returns the associated attribute value from the text attribute dimension, Pkg Type. The following table shows the value that would be returned:

Current Member	Return Value
100-10	Can
100-20	Can

Current Member	Return Value
100-30	Bottle
100	(empty string)
200-10	Bottle
200-20	Bottle
200-30	Bottle
200-40	Bottle
200	(empty string)
Product	(empty string)

For any member that does not have an associated attribute, an empty string is returned.

See Also

- [@ATTRIBUTEVAL](#)
- [@ATTRIBUTEVAL](#)

@ATTRIBUTEVAL

Returns, for the current member being calculated, the associated attribute value from the specified numeric or date attribute dimension.

Syntax

@ATTRIBUTEVAL (*attDimName*)

Parameter Description

attDimName Single dimension specification for a numeric or date attribute dimension.

Notes

- The @ATTRIBUTEVAL function works only with numeric and date attribute dimensions. To return values from text attribute dimensions, use the @ATTRIBUTESVAL function. To return values from Boolean attribute dimensions, use the @ATTRIBUTEVBVAL function. For more information on types of attributes, see the *Oracle Essbase Database Administrator's Guide*.
- Only level 0 members of attribute dimensions can be associated as attributes of members of a base dimension.
- If there is no attribute associated with the member being calculated, or if the attribute associated with the member is a text attribute, @ATTRIBUTEVAL returns #MISSING.
- When the @ATTRIBUTEVAL function is used with a date attribute dimension, Essbase converts the date string to the number of seconds elapsed since midnight, January 1, 1970.

Example

Example 1

The following example is based on the Sample Basic database:

```
"Profit Per Ounce" = Profit/@ATTRIBUTEVAL(@NAME(Ounces));
```

In this formula, for the current member being calculated, @ATTRIBUTEVAL returns the associated attribute from the Ounces numeric attribute dimension. For example, if the member being calculated is Cola and if the Ounces attribute value associated with Cola is 12, @ATTRIBUTEVAL returns 12. The value returned is then divided into Profit to yield Profit Per Ounce.

Note: The @NAME function is required to process the string “Ounces” before passing it to the @ATTRIBUTEVAL function.

This example produces the following report:

	Actual	Year	West
	Profit	Profit	Per Ounce
	=====		=====
Cola	4593		382.75

Example 2

The following MaxL execute calculation statement applies a formula to members that are 16 Oz products:

```
execute calculation
'Misc
( IF
  (@ATTRIBUTEVAL(Ounces) == 16)
  Misc = .5;
  ENDIF;
);'
on sample.basic;
```

See Also

- [@ATTRIBUTEVAL](#)
- [@ATTRIBUTESVAL](#)
- [@NAME](#)
- [@TODATE](#)

@AVG

Returns the average of all values in *expList*.

Syntax

```
@AVG (SKIPNONE | SKIPMISSING | SKIPZERO | SKIPBOTH, expList)
```

Parameter	Description
SKIPNONE	Includes all cells specified in the average operation regardless of their content.
SKIPMISSING	Excludes all values that are #MISSING in the average operation.
SKIPZERO	Excludes values of zero from the average calculation.
SKIPBOTH	Excludes all values of zero or #MISSING from the average calculation.
expList	Comma-delimited list of member names, variable names, functions, or numeric expressions. <i>expList</i> provides a list of numeric values across which the average is calculated.

Example

The following example is based on the Sample Basic database. The calculation averages the values for the individual states making up the western region and places the results in West:

```
FIX(Sales)
West=@AVG(SKIPNONE,California:Nevada);
ENDFIX
```

This example produces the following report:

	Sales		Jan	Actual
	Cola	Diet Cola	Cola	Caffeine Free Cola
	====	=====		=====
California	678	118		145
Oregon	160	140		150
Washington	130	190		#MI
Utah	130	190		170
Nevada	76	62		#MI
West	234.8	140		155

See Also

- [@AVGRANGE](#)

@AVGRANGE

Returns the average value of the specified member (*mbrName*) across the specified range (*XrangeList*).

Syntax

```
@AVGRANGE ( SKIPNONE | SKIPMISSING | SKIPZERO | SKIPBOTH, mbrName [, XrangeList])
```

Parameter	Description
SKIPNONE	Includes all cells specified in the average operation regardless of their content.
SKIPMISSING	Excludes all values that are #MISSING in the average operation.
SKIPZERO	Excludes values of zero from the average calculation.
SKIPBOTH	Excludes all values of zero or #MISSING from the average calculation.

Parameter	Description
mbrName	Any valid single member or member combination.
XrangeList	Optional. A valid member name, a comma-delimited list of member names, cross dimension members, or a member set function or range function (including @XRANGE) that returns a list of members from the same dimension. If <i>XrangeList</i> is not specified, Essbase uses the level 0 members from the dimension tagged as time.

Notes

The @AVGRANGE function accepts the @ATTRIBUTE member set function as a member range.

Example

The following example is based on the Sample Basic database. The calculation script determines the average sales of Colas in the West.

```
FIX(Sales)
West=@AVGRANGE(SKIPNONE, Sales, @CHILDREN(West));
ENDFIX
```

This example produces the following report:

	Sales	Colas	Actual
	Jan	Feb	Mar
	===	===	===
California	941	899	927
Oregon	450	412	395
Washington	320	362	377
Utah	490	488	476
Nevada	138	137	138
West	467.8	459.6	462.6

See Also

- [@AVG](#)

@BETWEEN

Returns a member set of all members whose name string value fall between, and are inclusive of, the two specified string tokens. Member names are evaluated alphanumerically.

This function can be used on unique and duplicate-name outlines.

Syntax

```
@BETWEEN (firstToken , secondToken, topMbrInHierarchy)
```

Parameter	Description
firstToken	First token string value with which to compare to members in the outline, starting with the member specified in <i>topMbr</i> .
secondToken	Second token string value with which to compare to members in the outline, starting with the member specified in <i>topMbr</i> .

Parameter	Description
topMbrInHierarchy	A fully qualified name of a member in the outline on which to base the member search. The specified member and its aliases, and all of its descendants, are included in the search. To search the entire outline, provide an empty string (" ") for this parameter. For example, @BETWEEN("200-10", "200-20", " ").

Example

The following example is based on the following duplicate-name outline:

```
Product
  100
    100-10
      100-10-10
    100-20
    100-30
  200
    200-10
    200-20
    200-30
  300
    300-10
    300-20
Diet
  100-10
    100-10-11
  200-10
  300-10
Bottle
  200-10
  300-20
```

```
@BETWEEN("200-10", "200-20", "Product")
```

Returns the members [200].[200-10], [200].[200-20], [Diet].[200-10], and [Bottle].[200-10].

See Also

- [@EQUAL](#)
- [@EXPAND](#)
- [@LIKE](#)
- [@MBRCOMPARE](#)
- [@MBRPARENT](#)
- [@NOTEQUAL](#)

@CALCMODE

Enables the choice of an execution mode of a formula. @CALCMODE can control two types of modes:

- Whether a formula is calculated in block calculation or cell calculation mode when calculating formulas that contain certain functions (in particular the @ISMBR function)

- Whether a formula assigned to a sparse member is calculated in bottom-up or top-down mode

Understanding Block Calculation and Cell Calculation Modes

Using block calculation mode, Essbase groups the cells within a block and simultaneously calculates the cells in each group. Block calculation mode is fast, but you must carefully consider data dependencies within the block to ensure that the resulting data is accurate.

Using cell calculation mode, Essbase calculates each cell sequentially, following the calculation order, which is based on the order of the dense dimensions in the outline. For more information on calculation order, see the *Oracle Essbase Database Administrator's Guide*.

Understanding Bottom-Up and Top-Down Calculation Modes

Essbase uses one of two methods to do a full calculation of an outline: bottom-up calculation (the default) or top-down calculation. If the outline contains a complex member formula, Essbase performs a top-down calculation for that member. When a formula is compiled, if the formula is to be calculated top-down, Essbase logs a message in the application log file.

For a bottom-up calculation, Essbase determines which existing data blocks need to be calculated before it calculates the database. Essbase then calculates only the blocks that need to be calculated during the full database calculation. The calculation begins with the lowest existing block number and works up through each subsequent block until the last existing block is reached.

In contrast, a top-down calculation calculates the formula on all potential datablocks with the member. A top-down calculation may be less efficient than a bottom-up calculation because more blocks may be calculated than is necessary. Although a top-down calculation is less efficient than a bottom-up calculation, in some cases top-down calculations are necessary to ensure that calculation results are correct. See [Example 4](#).

For more information about bottom-up and top-down calculation modes, see the *Oracle Essbase Database Administrator's Guide*.

Syntax

```
@CALCMODE (CELL|BLOCK|TOPDOWN|BOTTOMUP)
```

Parameter	Description
CELL	Turns on the cell calculation mode
BLOCK	Turns on the block calculation mode
TOPDOWN	Turns on the top-down calculation mode
BOTTOMUP	Turns on the bottom-up calculation mode

Notes

Cell and block modes are mutually exclusive. Top-down and bottom-up modes are mutually exclusive. Within one @CALCMODE specification, you can specify only one option. To specify both types of modes, perform the instruction twice; for example:

@CALCMODE (CELL)
@CALCMODE (TOPDOWN)

Knowing When Essbase uses Cell or Block Mode and Top-down or Bottom-up Mode

- When Essbase compiles a formula, it prints a message in the application log file explaining the mode of execution for the formula similar to the following message:

Formula on member Profit % will be executed in CELL and TOPDOWN mode.

When Essbase determines that the formula will be executed in block and bottom-up mode, no message is written in the application log file.

- In calculation scripts, @CALCMODE statements must be placed within parentheses and associated with a specific database member.
- By default, for a simple formula such as $A = B + C$, Essbase does a bottom-up calculation. A is calculated only if B or C exists in the database. The dependency of the formula on B and C is known before the calculation is started.

For a complex formula such as $A = B \rightarrow D + C \rightarrow D$, Essbase performs a top-down calculation because every possible combination of A must be examined to see whether $B \rightarrow D$ or $C \rightarrow D$ exists.

- By default, Essbase uses cell calculation mode for formulas containing:
 - @ANCEST
 - @CURRMBR
 - @ISMBR on a dense member
 - @MDANCESTVAL
 - @MDPARENTVAL
 - @MDSHIFT
 - @NEXT
 - @PARENT
 - @PARENTVAL
 - @PRIOR
 - @SANCESTVAL
 - @SPARENTVAL
 - @SHIFT

For all other formulas, Essbase uses block calculation mode by default.

You can also set CALCMODE BLOCK or CALCMODE BOTTOMUP at the Essbase server, application, or database level using the configuration setting CALCMODE.

Understanding Data Dependency Issues With Block Calculation Mode

Data dependency occurs if the accurate calculation of one or more members depends on another member or other on members being calculated previously. Most data dependency issues with block calculation mode occur when a formula contains IF ELSE or IF ELSEIF conditions.

However, data dependencies can occur in other formulas; for example, when using the @PRIOR function.

Data Dependency Issues With IF ELSE and IF ELSEIF

When Essbase uses block calculation mode to calculate a formula that contains IF ELSE or IF ELSEIF conditions, it separates the members being calculated into two groups. The first group contains the members that satisfy the IF condition. The second group contains the members that satisfy the ELSE or ELSEIF conditions.

Essbase simultaneously calculates the members in the first group before simultaneously calculating the members in the second group. See [Example 1](#).

If a formula contains data dependencies, ensure that the following conditions are met:

- Members on which the accurate calculation of other members depends are in the first group.
- Dependent members are in the second group.

If an IF condition has multiple ELSEIF conditions, Essbase evaluates each ELSEIF condition, placing the members that satisfy the ELSEIF condition in the first group and the members that satisfy subsequent ELSEIF or ELSE conditions in the second group. See [Example 2](#).

Understanding Other Data Dependency Issues

Data dependencies can occur in formulas that do not contain IF ELSE conditions. See [Example 3](#) for an example of data dependency in a formula containing the @PRIOR function.

You can also set CALCMODE BLOCK or CALCMODE BOTTOMUP at the Essbase server, application, or database level using the configuration setting CALCMODE.

Example

Example 1, Example 2, and Example 3 illustrate use of the BLOCK and CELL options of the @CALCMODE function. [Example 4](#) illustrates use of the BOTTOMUP and TOPDOWN options.

Example 1

Consider a database with two dense dimensions, Time and Accounts. The following formula is placed on the Budget Sales member of the Accounts dimension. Because this is a formula containing @ISMBR applied to a dense member (Budget Sales), by default Essbase uses cell calculation mode. Use the @CALCMODE(BLOCK) function to specify block calculation mode for this formula.

```
@CALCMODE (BLOCK) ;  
IF (@ISMBR (Feb) )  
    "Budget Sales" = 100 ;  
ELSE  
    "Budget Sales" = Feb + 10 ;
```

According to the above formula, we expect that if the member being calculated is Feb, the Budget Sales value is 100. If the member being calculated is not Feb, the Budget Sales value is 100+10 (the value for Feb + 10).

Assume that we load the values 10, 20, and 30 into the Budget Sales data block for Jan, Feb and Mar, as follows:

(axis)	Jan	Feb	Mar
Budget Sales	10	20	30

Using block calculation mode, Essbase calculates the members satisfying the IF condition first. In this example, Feb is the only member that satisfies the IF condition. After calculating Feb, Essbase calculates the members Jan and Mar. In this example, the results are as expected:

(axis)	Jan	Feb	Mar
Budget Sales	110	100	110

Example 2

Now consider the same database as in Example 1, but we place the following formula on the Budget Sales member of the Accounts dimension. As in Example 1, because this is a formula containing @ISMBR applied to a dense dimension member (Budget Sales), by default Essbase uses cell calculation mode. However, we use the @CALCMODE(BLOCK) function to specify the block calculation mode for this formula.

```
@CALCMODE (BLOCK) ;
IF (@ISMBR (Mar) )
    "Budget "->"Sales" =Feb+20;
ELSEIF (@ISMBR (Jan) )
    "Budget "->"Sales" =Feb+10;
ELSE
    "Budget "->"Sales" =100;
ENDIF
```

According to this formula, we want the Jan and Mar Budget Sales values to be calculated based on the Feb Budget Sales value, which is 100. We want to see the following results:

(axis)	Jan	Feb	Mar
Budget Sales	110	100	120

Assume that we load the values 10, 20, and 30 into the Budget Sales data block for Jan, Feb, and Mar, as follows:

(axis)	Jan	Feb	Mar
Budget Sales	10	20	30

Using block calculation mode, Essbase calculates the members satisfying the IF condition first, followed by the members satisfying the ELSEIF condition, followed by the members satisfying the ELSE condition. In this example, Essbase calculates the members in the following order: Mar, Jan, Feb. The results are not what we want, because the calculation of Jan and Mar is

dependent on the calculation of Feb and Feb is calculated after Jan and Mar. The inaccurate results are as follows:

(axis)	Jan	Feb	Mar
Budget Sales	30	100	40

To achieve the desired results, use the @CALCMODE(CELL) function.

Example 3

The following formula calculates the members Opening Inventory and Ending Inventory using the @PRIOR function. There is a data dependency between Opening Inventory and Ending Inventory. The formula is placed on the Opening Inventory member. The example shows the results for January, February, and March.

```
@CALCMODE (BLOCK)
"Opening Inventory"=@PRIOR("Ending Inventory")+10;
"Ending Inventory"="Opening Inventory";
```

Before the calculation, there is no data for these members (the data is #MISSING or #MI):

(axis)	Jan	Feb	Mar
Opening Inventory	#MI	#MI	#MI
Ending Inventory	#MI	#MI	#MI

Using block calculation mode, Essbase calculates the members simultaneously, taking the previous month's Ending Inventory #MISSING value as 0 for all member combinations and adding 10. This is not the desired result.

(axis)	Jan	Feb	Mar
Opening Inventory	10	10	10
Ending Inventory	10	10	10

The following formula on the Opening Inventory member causes Essbase to use cell calculation mode (the default for formulas containing the @PRIOR function):

```
"Opening Inventory"=@PRIOR("Ending Inventory")+10;
"Ending Inventory"="Opening Inventory";
```

The results are as follows:

(axis)	Jan	Feb	Mar
Opening Inventory	10	20	30
Ending Inventory	10	20	30

Example 4

Depending on the formula and the structure of the data, calculating a formula top-down versus bottom-up may involve two issues: performance (reflecting the number of calculations that must be made) and accuracy. This example compares calculation results to illustrate both of these issues.

Before the calculation, assume that Actual and Budget are members of a dense dimension and they contain the following data:

(axis)	Cola	New York Sales
(axis)	Actual	Budget
Jan	#MISSING	50
Feb	200	#MISSING
Mar	400	450

The following formula is calculated bottom-up.

```
Budget (  
    @CALCMODE (BOTTOMUP) ;  
    Budget=Actual*1.10;  
)
```

In a bottom-up calculation, Essbase executes formulas only from existing data blocks. Therefore, only two values—Jan and Mar—are calculated, based on existing combinations of Budget.

(axis)	Cola	New York Sales	(Comment)
(axis)	Actual	Budget	
Jan	#MISSING	#MISSING	(#MISSING*1.10)
Feb	200	#MISSING	(No calculation is performed)
Mar	400	440	(400*1.10)

The following formula is calculated top-down.

```
Budget (  
    @CALCMODE (TOPDOWN) ;  
    Budget=Actual*1.10;  
)
```

In a top-down calculation, Essbase materializes every potential data block that is relevant to the calculation, and executes formulas in those blocks. Therefore, all three values—Jan, Feb, and Mar—are calculated, based on all potential combinations of Budget. The results are:

(axis)	Cola	New York Sales	(Comment)
(axis)	Actual	Budget	
Jan	#MISSING	#MISSING	(#MISSING*1.10)
Feb	200	220	(200*1.10)
Mar	400	440	(400*1.10)

See Also

- [@WITHATTR](#)
- [“CALCMODE” on page 402](#)

@CHILDREN

Returns all children of the specified member, excluding the specified member. This member set function can be used as a parameter of another function, where that parameter is a list of members.

Syntax

@CHILDREN (*mbrName*)

Parameter Description

mbrName Any valid single member name or member combination, or a function that returns a single member or member combination.

Notes

Essbase sorts the child members in ascending order. Using Sample Basic as an example, if you specify 100 for *mbrName*, Essbase returns 100-10, 100-20, 100-30 (in that order). This order is important to consider when you use the @CHILDREN member set function with certain forecasting and statistical functions.

Example

In the Sample Basic database:

```
@CHILDREN(Market)
```

returns East, West, South, and Central (in that order).

```
@CHILDREN(Margin)
```

returns Sales and COGS (in that order).

See Also

- [@ICHILDREN](#)
- [@ISCHILD](#)
- [@ANCESTORS](#)
- [@DESCENDANTS](#)

- [@SIBLINGS](#)

@COMPOUND

Compiles the proceeds of a compound interest calculation. The calculation is based on the balances of the specified member at the specified rate across the specified range.

Syntax

```
@COMPOUND (balanceMbr, rateMbrConst [, rangeList])
```

Parameter Description

balanceMbr	Single member specification representing the beginning balance across a range of periods. The input can be either one deposit or a series of deposits. If <i>balanceMbr</i> is a constant, then Essbase assumes <i>balanceMbr</i> to be a single deposit in the first member of <i>rangeList</i> . This is equivalent to entering the constant value in the first member in the <i>rangeList</i> followed by zeros. The function keeps track of each deposit separately, but returns a composite value. If <i>balanceMbr</i> is a member, or a range, then it is assumed to be a series of deposits.
rateMbrConst	Single member specification, variable name, or numeric expression in decimal form. This represents the interest rate per time period specified in the <i>rangeList</i> . If your interest is compounded monthly, this value would be the annual interest rate divided by 12.
rangeList	Optional. A valid member name, a comma-delimited list of member names, member set functions, and range functions from the dimension tagged as Time. If <i>rangeList</i> is not specified, Essbase uses the level 0 members from the dimension tagged as Time. <i>rangeList</i> represents the range over which the interest is compounded. The last value in the range is the total compounded interest for that range.

Notes

Financial functions never return a value; rather, they calculate a series of values internally based on the range specified.

Example

The following example determines the compound interest of a series of deposits, based on a credit rate of 0.0525, across a series of fiscal years:

```
"Compound Interest"=@COMPOUND(Deposit,"Credit Rate",FY1998:FY2001,FY2002);
```

This example produces the following report:

	FY1998	FY1999	FY2000	FY2001	FY2002
	=====	=====	=====	=====	=====
Credit Rate	0.0525	0.0525	0.0525	0.0525	0.0525
Compound Interest	0	105	110.5125	273.8144	288.1897
Deposit	0	2,000	0	3,000	0

See Also

- [@INTEREST](#)

@COMPOUNDGROWTH

Calculates a series of values that represents a compound growth of values (the first nonzero value in the specified member across the specified range of members) across time.

The growth factor is calculated by multiplying the growth rate in the current time period by the previous period's result, yielding a compounded value. You can change the growth rate from period to period by placing a nonzero value in the current period's *rateMbrConst* cell.

Syntax

```
@COMPOUNDGROWTH (principalMbr, rateMbrConst [, rangeList])
```

Parameter	Description
-----------	-------------

<i>principalMbr</i>	Member specification representing the initial value to be compounded. The input line must be a single deposit.
<i>rateMbrConst</i>	Single member specification, variable name, or expression which provides a constant value. This value can change across <i>rangeList</i> , making the new value be the new compound rate. If the value in the current period is zero, the compound rate is equal to zero, and the principal does not change.
<i>rangeList</i>	Optional. A valid member name, a comma-delimited list of member names, member set functions, and range functions from the dimension tagged as Time. If <i>rangeList</i> is not specified, Essbase uses the level 0 members from the dimension tagged as Time.

Notes

Financial functions never return a value; rather, they calculate a series of values internally based on the range specified.

Example

The following example determines the compound growth of Principal Amount based on Growth Rate across a series of fiscal years.

```
"Compound Growth"=@COMPOUNDGROWTH("Principal Amount",  
  "Growth Rate",FY1998:FY2003);
```

This example produces the following report:

	FY1998	FY1999	FY2000	FY2001	FY2002	FY2003
Principal Amount	2,000	2,000	2,000	3,000	2,500	-500
Growth Rate	0.0525	0	0	0	0	0
Compound Growth	2,105	2,105	2,105	2,105	2,105	2,105

See Also

- [@GROWTH](#)

@CONCATENATE

Returns one character string that is the result of appending one character string (*String2*) to the end of another character string (*String1*).

The @CONCATENATE function can be nested to concatenate more than two strings (See [Example 2 \(@CONCATENATE\)](#)).

Syntax

```
@CONCATENATE (String1, String2)
```

Parameter Description

String1 A string or a function that returns a string

String2 A string or a function that returns a string

Notes

- To use a member name as a character string, use @NAME with the member name.
- To use the resulting character string as a member name, use @MEMBER with the @CONCATENATE statement; for example,

```
@MEMBER (@CONCATENATE ("2000_", QTR1));
```

Example

The following examples are based on the Sample Basic database:

Example 1 (@CONCATENATE)

The following function statement puts the string Item in front of the name of the member currently being processed in the Product dimension; for example, if the current member being calculated is 100-10, the result is Item100-10:

```
@CONCATENATE ("Item", @NAME (@CURRMBR (Product)))
```

Example 2 (@CONCATENATE)

To concatenate more than two strings, you can nest multiple instances of the @CONCATENATE function. The following function statement returns string values starting with the current member of the Year dimension, followed by an underscore, followed by the current member of the Measures dimension; for example, if the current members being calculated are Qtr1 and Sales, the result is Qtr1_Sales:

```
@CONCATENATE (@NAME (@CURRMBR (Year)), @CONCATENATE ("_", @NAME (@CURRMBR (Measures))))
```

See Also

- [@SUBSTRING](#)
- [@MEMBER](#)
- [@NAME](#)

@CORRELATION

Returns the correlation coefficient between two parallel data sets (*expList1* and *expList2*). The correlation coefficient determines the relationship between two data sets.

Syntax

@CORRELATION (SKIPNONE | SKIPMISSING | SKIPZERO | SKIPBOTH, *expList1*, *expList2*)

Parameter	Description
SKIPNONE	Includes all cells specified in <i>expList1</i> and <i>expList2</i> , regardless of their content, during calculation of the correlation coefficient.
SKIPMISSING	Excludes all #MISSING values from <i>expList1</i> and <i>expList2</i> during calculation of the correlation coefficient.
SKIPZERO	Excludes all zero (0) values from <i>expList1</i> and <i>expList2</i> during calculation of the correlation coefficient.
SKIPBOTH	Excludes all zero (0) values and #MISSING values from <i>expList1</i> and <i>expList2</i> during calculation of the correlation coefficient.
<i>expList1</i>	The first list of member specifications, variable names, functions, or other numeric expressions.
<i>expList2</i>	The second list of member specifications, variable names, functions, or other numeric expressions.

Notes

- For complete information about using the @RANGE function, see [@RANGE](#).
- The *expList1* and *expList2* parameters must have the same number of data points. If *expList1* and *expList2* have different numbers of data points, @CORRELATION returns #MISSING.
- The @CORRELATION function returns #MISSING if *expList1* and *expList2* (1) are empty, (2) contain only #MISSING values, or (3) have a standard deviation of 0 (all values are constant).
- The @CORRELATION function treats #MISSING values as zero (0) values, unless SKIPMISSING or SKIPBOTH is specified. If a value in *expList1* is #MISSING, and SKIPMISSING is specified, the value's corresponding value in *expList2* is treated as #MISSING. (That is, both values are deleted before calculation.) SKIPZERO and SKIPBOTH work similarly.
- The @CORRELATION function returns values from -1 to 1.
- If you use a member set function to generate a member list for this function (for example, @SIBLINGS), to ensure correct results, consider the order in which Essbase sorts the generated member list. For more information, see the *Oracle Essbase Technical Reference* topic for the member set function you are using.
- The equation for the correlation coefficient is:

$$\rho_{x,y} = \frac{\text{Cov}(X,Y)}{\sigma_x * \sigma_y}$$

so that

$$-1 \leq \rho_{x,y} \leq 1$$

and

$$\text{Cov}(X,Y) = \frac{1}{n} \sum_{i=1}^n (x_i - \mu_x)(y_i - \mu_y)$$

σ_x stands for the standard deviation of $X = \{x_i\}_{i=1}^n$

σ_y stands for the standard deviation of $Y = \{y_i\}_{i=1}^n$

Example

The following example is based on the Sample Basic database. Assume that the Measures dimension contains an additional member, Sales Correl. The calculation script calculates the correlation coefficient for a set of members (Sales for the children of Qtr1 and Qtr2). Because the calculation script fixes on Jun, the results are placed in Sales Correl->Jun.

This example uses the [@RANGE](#) function to generate *expList1* and *expList2*:

```
FIX(June)
"Sales Correl"=@CORRELATION(SKIPNONE,
@RANGE(Sales,@CHILDREN(Qtr1)),@RANGE(Sales,@CHILDREN(Qtr2)));
ENDFIX
```

This example produces the following report:

	Colas	Actual	New York
	Sales	Sales	Correl
	=====	=====	=====
Jan	678		#MI
Feb	645		#MI
Mar	675		#MI
Apr	712		#MI
May	756		#MI
Jun	890	0.200368468	

See Also

- [@RANGE](#)

@COUNT

Returns the number of data values in the specified data set (*expList*).

Syntax

@COUNT (SKIPNONE | SKIPMISSING | SKIPZERO | SKIPBOTH, *expList*)

Parameter	Description
-----------	-------------

SKIPNONE	Includes all cells specified in <i>expList</i> , regardless of their content, during calculation of the count.
SKIPMISSING	Excludes all #MISSING values from <i>expList</i> during calculation of the count.
SKIPZERO	Excludes all zero (0) values from <i>expList</i> during calculation of the count.
SKIPBOTH	Excludes all zero (0) values and #MISSING values from <i>expList</i> during calculation of the count.
<i>expList</i>	Comma-delimited list of member specifications, variable names, functions, or numeric expressions.

Notes

The @COUNT function always returns an integer greater than or equal to 0.

Example

The following example is based on the Sample Basic database. Assume that the Measures dimension contains an additional member, Prod Count. This example calculates the count of all products for which a data value exists and uses the @RANGE function to generate *expList*:

```
FIX(Product)
"Prod Count" = @COUNT(SKIPMISSING, @RANGE(Sales, @CHILDREN(Product)));
ENDFIX
```

This example produces the following report. Since SKIPMISSING is specified in the calculation script, the #MI values for Diet Drinks are skipped during the product count.

		Jan	New York
		Actual	Budget
		=====	=====
Sales	Colas	678	640
	Root Beer	551	530
	Cream Soda	663	510
	Fruit Soda	587	620
	Diet Drinks	#MI	#MI
	Product	2479	2300
Prod Count	Product	4	4

See Also

- [@RANGE](#)

@CURGEN

Returns the generation number of the current member combination for the specified dimension. This number represents the number of members separating the current member from the top-most member of the dimension.

Syntax

@CURGEN (*dimName*)

Parameter Description

dimName Single dimension name specification. *dimName* must be the name of the top-most member of the dimension. It cannot be another member name from within the dimension.

Notes

- If the current member of the specified dimension is an implied share member, the member generation returned is the same generation as the stored member. For example, in Sample Basic, Inventory, a member of the Measures dimension, is an implied share member:

```
Inventory
  Opening Inventory (+)
  Additions (~)
  Ending Inventory (~)
```

The generation value of Inventory is the same as the stored member under it, Opening Inventory. For this example, Opening Inventory is at generation 3. When Inventory is the current member @CURGEN(Measures) returns generation 3.

- For further discussion on levels, please refer to the *Oracle Essbase Database Administrator's Guide*.

Example

Given the following database structure:

```
Year
  Qtr1
    Jan, Feb, Mar
  Qtr2
    Apr, May, Jun
  Qtr3
    Jul, Aug, Sep
  Qtr4
    Oct, Nov, Dec
```

@CURGEN provides the following results for the members shown:

Formula	Current Member	Value
Position = @CURGEN(Year);	Year	1
Position = @CURGEN(Year);	Qtr2	2
Position = @CURGEN(Year);	Oct	3

See Also

- [@CURLEV](#)
- [@GEN](#)

@CURLEV

Returns the level number of the current member combination for the specified dimension. This number represents the number of members that separates the current member from its bottom-most descendant.

Syntax

@CURLEV (*dimName*)

Parameter Description

dimName Single dimension name specification. *dimName* must be the name of the top-most member of the dimension. It cannot be another member name from within the dimension.

Notes

- If the current member of the specified dimension is an implied share member, the member level returned is the same level as the stored member. For example, in Sample Basic, Inventory, a member of the Measures dimension, is an implied share member:

```
Inventory
  Opening Inventory (+)
  Additions (~)
  Ending Inventory (~)
```

The value of Inventory results only from the value of Opening Inventory.

When Inventory is the current member @CURLEV (Measures) returns level 0.

- For further discussion on levels, please refer to the *Oracle Essbase Database Administrator's Guide*.

Example

Given the following database structure:

```
Year
  Qtr1
    Jan, Feb, Mar
  Qtr2
    Apr, May, Jun
  Qtr3
    Jul, Aug, Sep
  Qtr4
    Oct, Nov, Dec
```

@CURLEV provides the following results for the members shown:

Formula	Current Member	Value
Position = @CURLEV(Year);	Year	2
Position = @CURLEV(Year);	Qtr3	1
Position = @CURLEV(Year);	Aug	0

See Also

- [@CURGEN](#)
- [@LEV](#)

@CURRMBR

Returns the member that is currently being calculated in the specified dimension (*dimName*). This function can be used as a parameter of another function, where that parameter is a single member or a list of members.

Syntax

```
@CURRMBR (dimName)
```

Parameter Description

dimName A single dimension name.

Notes

- You cannot use the @CURRMBR function in a FIX statement.
- You cannot use the @CURRMBR function on the left-hand side of a formula.
- The time required for retrieval and calculation may be significantly longer if this function is in a formula attached to a member tagged as Dynamic Calc or Dynamic Calc and Store.

Example

In the Sample Basic database,

```
@CURRMBR(Year);
```

returns Jan if the current member of Year being calculated is Jan.

As a more complex example, consider the following formula in the context of the Sample Basic database. Assume that the Measures dimension contains an additional member, Average Sales.

```
"Average Sales"  
(IF(@ISLEV(Product,0))  
Sales;  
ELSE  
@AVGRANGE(SKIPNONE,Sales,@CHILDREN(@CURRMBR(Product))));  
ENDIF);
```

This formula populates each upper-level member of the Product dimension (100, 200) at Average Sales. To calculate Average Sales, the Sales values for the level 0 members of Product are averaged and placed in their respective parent members. The Average Sales values for the level 0 Product members are the same as the Sales values, as specified by the IF statement in the calculation script.

This example produces the following report:

	Jan	New York	Actual
	Sales		Average Sales
	=====		=====
100-10	5		5
100-20	10		10
100-30	15		15
100	30		10
200-10	20		20

200-20	25	25
200-30	30	30
200-40	35	35
200	110	27.5
300	#MI	#MI
400	#MI	#MI
Diet	35	11.67
Product	140	35

See Also

- [@CURRMBRRANGE](#)

@CURRMBRRANGE

Generates a member list that is based on the relative position of the current member being calculated.

Syntax

```
@CURRMBRRANGE (dimName, {GEN|LEV}, genLevNum, [startOffset], [endOffset])
```

Parameter Description

dimName	Name of the dimension for which you want to return the range list.
GEN LEV	Defines whether the range list to be returned is based on a generation or a level within the dimension.
genLevNum	Integer value that defines the absolute generation or level number of the range list to be returned.
startOffset	Defines the first member in the range to be returned. <ul style="list-style-type: none"> • A null value returns the first member of the specified <i>genLevNum</i>. • An integer value returns the member name relative to the current member being calculated. • A negative value specifies a member prior to the current member being calculated in the dimension. • A value of 0 returns the name of the member currently being calculated. • A positive value specifies a member after the current member being calculated in the dimension.
endOffset	Defines the last member in the range to be returned. <ul style="list-style-type: none"> • A null value returns the last member of the specified <i>genLevNum</i>. • An integer value returns the member name relative to the current member being calculated. • A negative value specifies a member prior to the current member being calculated in the dimension. • A value of 0 returns the name of the member currently being calculated. • A positive value specifies a member after the current member being calculated in the dimension.

Notes

- You cannot use the @CURRMBRRANGE function in a FIX statement.
- The first three parameters of this function (*dimName*, {GEN|LEV}, *genLevNum*) provide a member range list. The *startOffset* and *endOffset* parameters create a subset of this list. For example, consider the following syntax in the context of the Sample Basic database:

```
@CURRMBRRANGE (Year, LEV, 0, -1, 1)
```

In this example, the full range list contains the level 0 members of the Year dimension (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec). If the current member being calculated in the Year dimension is Jan, the *startOffset* and *endOffset* parameters reduce this list to (Jan, Feb). Since there is no member prior to Jan in the full range list, only two members are returned: Jan itself and the member after it, Feb. If the current member being calculated is Feb, the subset list would include three members: Jan, Feb, Mar.

- Currently, this function can be used only within range and financial functions, such as @AVGRANGE, @MAXRANGE, @COMPOUND, and @SHIFT.

Example

Example 1

Average Inventory is calculated by summing opening inventories from the first month of the year to the current period plus one period, and dividing the result by the number of periods to date plus one period. This calculation is accomplished by defining the @CURRMBRRANGE function within the *rangeList* parameter of the @AVGRANGE function.

```
"Average Inventory" = @AVGRANGE(SKIPNONE, "Opening Inventory",
@CURRMBRRANGE(Year, LEV, 0, , 1));
```

This example produces the following result:

	Jan	Feb	Mar	Apr	Nov	Dec
Opening Inventory	100	110	120	130 . . .	200	210
Average Inventory	105	110	115	120 . . .	155	155

Since a null value is specified for *startOffset*, the average operations always begin at the first member of the range list, Jan. The *endOffset* parameter, 1, specifies that the member after the current member being calculated is included in each average operation. So, for Average Inventory->Jan, the values for Jan and Feb are averaged; for <Average Inventory->Feb, the values for Jan, Feb, and Mar are averaged; and so on. The values for Nov and Dec are the same since there is no member after Dec in the range list.

Example 2

Inventory Turnover is calculated by summing period-to-date Sales and dividing the result by the Average Inventory.

```
Turnover = @SUMRANGE(Sales,@CURRMBRRANGE(Year, LEV, 0, , 0))/"Average Inventory"
```

which produces the following result:

	Jan	Feb	Mar	Apr
Average Inventory	110	116.7	122.5	126
Sales	40	44	48	52
Turnover	0.36	0.72	1.08	1.46

Example 3

Consider the following formula:

```
@CURRMBRRANGE(Year, LEV, @CURLEV("Year"), -1, 1)
```

The full range list contains the members of the Year dimension at a particular level. The level is determined by taking the level of the current member being calculated. For example, if the current member being calculated is Jan, the full range list contains all level 0 members of Year

dimension (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec). The *startOffset* and *endOffset* parameters reduce this list to (Jan, Feb). As there is no member prior to Jan in the full range list, only two members are returned: Jan and Feb. If the current member being calculated is Feb, the subset list includes three members: Jan, Feb, Mar.

Note: The usage demonstrated by this example would require RTDEPCALCOPTIMIZE to be set to FALSE.

See Also

- [RTDEPCALCOPTIMIZE](#)
- [@CURGEN](#)
- [@CURLEV](#)

@DATEDIFF

Returns the difference (number) between two input dates in terms of the specified date-parts, following a standard Gregorian calendar.

Syntax

```
@DATEDIFF ( date1, date2, date_part )
```

Parameter Description

date1 A number representing the input date between January 1, 1970 and Dec 31, 2037. The number is the number of seconds elapsed since midnight, January 1, 1970. To retrieve this number, use any of the following functions: @TODAY, @TODATEEX, @DATEROLL.

Date-time attribute properties of a member can also be used to retrieve this number. For example, @AttributeVal ("Intro Date"); returns the product introduction date for the current product in context.

date2 A second input date. See *date1*.

date_part Defined using the following rule:

```
date_part_ex ::= DP_YEAR | DP_QUARTER | DP_MONTH | DP_WEEK | DP_DAY |  
DP_DAYOFYEAR | DP_WEEKDAY
```

Defined time components as per the standard calendar:

- DP_YEAR - Year of the input date.
- DP_QUARTER - Quarter of the input date.
- DP_MONTH - Month of the input date.
- DP_WEEK - Week of the input date.
- DP_DAY - Day of the input date.

Notes

Based on the input *date_part*, the difference between the two input dates is counted in terms of time component specified.

Example: For input dates June 14, 2005 and Oct 10, 2006,

- DP_YEAR returns the difference in the year component. (2006 - 2005 = 1)
- DP_QUARTER returns the distance between the quarters capturing the input dates. (Quarter 4, 2006 - Quarter 2, 2005 = 6)
- DP_MONTH returns the distance between the months capturing the input dates. (Oct 2006 - June 2005 = 16)
- DP_WEEK returns the distance between the weeks capturing the input dates. Each Standard calendar week is defined to start on Sunday and it spans 7 days. (Oct 10, 2006 - June 14, 2005 = 69)
- DP_DAY returns the difference between the input dates in terms of days. (483 days)

Example

Assume the outline has two date type members, MyDate1 and MyDate2.

```
Profit=@DateDiff(MyDate1, MyDate2, DP_WEEK);
Profit=@DatePart(MyDate1, DP_YEAR);
MyDate2=@DateRoll(MyDate1, DP_MONTH), 10);
```

See Also

- [@DATEPART](#)
- [@DATEROLL](#)
- [@FORMATDATE](#)
- [@TODATEEX](#)
- [@TODAY](#)

@DATEPART

This function returns the Year/Quarter/Month/Week/Day/DayOfYear/Weekday as a number, given the input date and a date part, following the standard Gregorian calendar.

Syntax

```
@DATEPART ( date, date_part_ex )
```

Parameter	Description
date	A number representing the input date between January 1, 1970 and Dec 31, 2037. The number is the number of seconds elapsed since midnight, January 1, 1970. To retrieve this number, use any of the following functions: @TODAY, @TODATEEX, @DATEROLL. Date-time attribute properties of a member can also be used to retrieve this number. For example, @AttributeVal("Intro Date"); returns the product introduction date for the current product in context.

Parameter	Description
-----------	-------------

date_part_ex Defined using the following rule:

```
date_part_ex ::= DP_YEAR | DP_QUARTER | DP_MONTH | DP_WEEK | DP_DAY |  
DP_DAYOFYEAR | DP_WEEKDAY
```

Defined time components as per the standard calendar:

- DP_YEAR - Year of the input date.
- DP_QUARTER - Quarter of the input date.
- DP_MONTH - Month of the input date.
- DP_WEEK - Week of the input date.
- DP_DAY - Day of the input date.

Notes

Based on the requested time component, the output is as follows:

- DP_YEAR returns the year of the input date in *yyyy* format.
- DP_QUARTER returns the quarter of the year (1 to 4) for the input date.
- DP_MONTH returns the month of the year (1 to 12) for the input date.
- DP_WEEK returns the week of the year for the input date (1 to 54).
- DP_WEEKDAY returns the week day of the input date. (1 - Sunday, 2 - Monday, ... 6 - Saturday).
- DP_DAYOFYEAR returns the day of the year numbering (1 to 366).
- DP_DAY returns the day of the month (1 to 31).

Example: For June 14, 2005,

DP_YEAR returns 2005 (the year member, in *yyyy* format).

DP_QUARTER returns 2 (Second quarter of the year)

DP_MONTH returns 6 (Sixth month of the year)

DP_WEEK returns 24 (24th week of the year)

DP_WEEKDAY returns 4 (for Wednesday. Sunday = 1)

DP_DAYOFYEAR returns 165 (165th day of the year)

DP_DAY returns 14 (14th day of the month)

Example

Assume the outline has two date type members, MyDate1 and MyDate2.

```
Profit=@DateDiff(MyDate1, MyDate2, DP_WEEK);  
Profit=@DatePart(MyDate1, DP_YEAR);  
MyDate2=@DateRoll(MyDate1, DP_MONTH), 10);
```

See Also

- [@DATEDIFF](#)
- [@DATEROLL](#)
- [@FORMATDATE](#)
- [@TODATEEX](#)
- [@TODAY](#)

@DATEROLL

To the given date, rolls (adds or subtracts) a number of specific time intervals, returning another date. This function assumes a standard Gregorian calendar.

Syntax

```
@DATEROLL ( date, date_part, number )
```

Parameter Description

date A number representing the date between January 1, 1970 and Dec 31, 2037. The number is the number of seconds elapsed since midnight, January 1, 1970. To retrieve this number, use either of the following functions: @TODAY, @TODATEEX.

Date-time attribute properties of a member can also be used to retrieve this number. For example, @AttributeVal ("Intro Date"); returns the product introduction date for the current product in context.

date_part Defined using the following rule:

```
date_part_ex ::= DP_YEAR | DP_QUARTER | DP_MONTH | DP_WEEK | DP_DAY |  
DP_DAYOFYEAR | DP_WEEKDAY
```

Defined time components as per the standard calendar:

- DP_YEAR - Year of the input date.
- DP_QUARTER - Quarter of the input date.
- DP_MONTH - Month of the input date.
- DP_WEEK - Week of the input date.
- DP_DAY - Day of the input date.

number Number of time intervals to add or subtract.

Notes

Based on input *date_part* and dateroll *number*, the date is moved forward or backward in time.

Example: For input date June 14, 2005 and input dateroll number 5,

- DP_YEAR adds 5 years to the input date. (June 14, 2010)
- DP_QUARTER adds 5 quarters to the input date. (June 14, 2005 + 5 quarters = June 14, 2005 + 15 months = Sept 14, 2006)
- DP_MONTH adds 5 months to the input date (June 14, 2005 + 5 months = Nov 14, 2005)
- DP_WEEK adds 5 weeks to the input date (June 14, 2005 + 5 weeks = June 14, 2005 + 35 days = July 19, 2005)

- DP_DAY adds 5 days to the input date. (June 14, 2005 + 5 days = June 19, 2005)

Example

Assume the outline has two date type members, MyDate1 and MyDate2.

```
Profit=@DateDiff(MyDate1, MyDate2, DP_WEEK);
Profit=@DatePart(MyDate1, DP_YEAR);
MyDate2=@DateRoll(MyDate1, DP_MONTH, 10);
```

See Also

- [@DATEDIFF](#)
- [@DATEPART](#)
- [@FORMATDATE](#)
- [@TODATEEX](#)
- [@TODAY](#)

@DECLINE

Calculates the depreciation of an asset for the specified period using the declining balance method. The factor by which the declining balance depreciates the assets is specified using *factorMbrConst*. For example, to calculate a double declining balance, set *factorMbrConst* to 2.

Syntax

```
@DECLINE (costMbr, salvageMbrConst, lifeMbrConst, factorMbrConst [, rangeList])
```

Parameter	Description
costMbr	Single member specification representing the starting values of the assets. More than one asset can be input and depreciated across the specified range. The function calculates each asset separately.
salvageMbrConst	Single member specification, variable name, or numeric expression that provides a constant value. This value represents the value of the asset at the end of the depreciation.
lifeMbrConst	Single member specification, variable name, or numeric expression that provides a constant value. The value represents the number of periods over which the asset is depreciated.
factorMbrConst	Single member specification, variable name, or numeric expression that provides a constant value. The value represents the factor by which the asset is depreciated.
rangeList	Optional. A valid member name, a comma-delimited list of member names, member set functions, and range functions from the dimension tagged as Time. If <i>rangeList</i> is not specified, Essbase uses the level 0 members from the dimension tagged as Time. The range represents the periods over which the function is calculated. More than one asset can be depreciated.

Notes

Financial functions never return a value; rather, they calculate a series of values internally based on the range specified.

Example

The following example calculates the depreciation of Asset for the specified series of fiscal years.


```
"Decline Dep" = @DECLINE(Asset,Residual,Life,2,FY2000:FY2001,FY2002,FY2003);
```

This example produces the following report:

	FY2000	FY2001	FY2002	FY2003
	=====	=====	=====	=====
Asset	9,000	0	0	0
Residual	750	0	0	0
Life	5	0	0	0
Decline Dep	3,600	2,160	1,296	778

See Also

- [@SLN](#)
- [@GROWTH](#)

@DESCENDANTS

Returns all descendants of the specified member, or those down to the specified generation or level. This function excludes the specified member.

Syntax

```
@DESCENDANTS (mbrName [, genLevNum| genLevName])
```

Parameter	Description
-----------	-------------

mbrName	Any valid single member name or member combination, or a function that returns a single member or member combination.
---------	-----------------------------------------------------------------------------------------------------------------------

genLevNum	Optional. An integer value that defines the absolute generation or level number down to which to select the members. A positive integer defines a generation number. A value of 0 or a negative integer defines a level number.
-----------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

genLevName	Optional. Level name or generation name down to which to select the members.
------------	------------------------------------------------------------------------------

Notes

- You can use this member set function as a parameter of another function, where that parameter is a list of members.
- Essbase sorts the generated list of members starting with the nearest descendant of the member, followed by the next nearest descendant of the member, and so on. In the Sample.Basic database, if you specify `@DESCENDANTS(100)`, Essbase returns 100-10, 100-20, 100-30 (in that order). This order is important to consider when you use the `@DESCENDANTS` member set function with certain forecasting and statistical functions.
- You can use [@IDESCENDANTS](#), to include the specified member.
- You can use [@RDESCENDANTS](#) and [@IRDESCENDANTS](#) to include descendants of shared members.

Example

In the Sample Basic database:

```
@DESCENDANTS (East)
```

returns New York, Massachusetts, Florida, Connecticut, and New Hampshire (in that order).

```
@DESCENDANTS (Profit)
```

returns Margin, Sales, COGS, Total Expenses, Marketing, Payroll, and Misc (in that order).

```
@DESCENDANTS (Market, 2)
```

returns East, West, South, and Central (in that order).

```
@DESCENDANTS (Diet, 0)
```

returns 100-20, 200-20, and 300-30 (in that order).

See Also

- [@IDESCENDANTS](#)
- [@LDESCENDANTS](#)
- [@ILDESCENDANTS](#)
- [@RDESCENDANTS](#)
- [@IRDESCENDANTS](#)
- [@ISDESC](#)
- [@ANCESTORS](#)
- [@CHILDREN](#)
- [@SIBLINGS](#)

@DISCOUNT

Calculates a value discounted by the specified rate, from the first period of the range to the period in which the amount to discount is found. The answer is returned in the same period. More than one value can be discounted simultaneously in this manner.

Syntax

```
@DISCOUNT (cashMbr, rateMbrConst [, rangeList])
```

Parameter	Description
cashMbr	Member specification representing the value you want to discount from the last period in <i>rangeList</i> to the current period.
rateMbrConst	Member specification, variable name, or numeric expression which provides a constant value. The value represents the rate per period which <i>cashMbr</i> is discounted. It is a decimal value, not a percent.
rangeList	Optional. A valid member name, a comma-delimited list of member names, member set functions, and range functions. If <i>rangeList</i> is not specified, Essbase uses the level 0 members from the dimension tagged as Time.

Notes

Financial functions never return a value; rather, they calculate a series of values internally based on the range specified.

Example

The following example discounts the values in Cash by the rates in Credit Rate and places the results in Discount Amount for each fiscal year.

```
"Discount Amount" = @DISCOUNT(Cash, "Credit Rate", FY1999:FY2002, FY2003);
```

This example produces the following report:

	FY1999	FY2000	FY2001	FY2002	FY2003
	=====	=====	=====	=====	=====
Cash	0.00	0.00	1000.00	1000.00	0.00
Credit Rate	0.00	0.00	0.05	0.05	0.00
Discount Amount	#MI	#MI	863.84	822.70	#MI

@ENUMVALUE

Returns the internal numeric value for a text value in a text list.

Syntax

```
@ENUMVALUE (enum_string)
```

Parameter	Description
-----------	-------------

enum_string String of the format *text_list_name.char_string_literal*, where:

- *text_list_name* is the name of a text list, or of a member that is associated with a text list.
- *char_string_literal* is one of the text values represented in the text list.

Example

The following example is based on a variation of ASOSamp.Sample. Assume there is a text list named CustSatRatings, in which text values are mapped to numeric IDs as follows: Good=1, Average=2, Poor=3.

```
@ENUMVALUE(CustSatRatings, "Good");
```

returns 1.

@EQUAL

Returns a member set of member names that match the specified token name.

This function can be used on unique and duplicate-name outlines.

Syntax

```
@EQUAL (tokenName, topMbrinHierarchy)
```

Parameter	Description
-----------	-------------

tokenName Token string value, representing the name of a member, with which to compare to members in the outline, starting with member specified in *topMbrinHierarchy*. The specified token name must not be qualified for duplicate members.

Parameter	Description
topMbrinHierarchy	A fully qualified name of a member in the outline on which to base the member search. The specified member and its aliases, and all of its descendants, are included in the search. To search the entire outline, provide an empty string (" ") for this parameter. For example, @EQUAL("100-10", " ").

Example

The following examples are based on the following duplicate-name outline:

```
Product
  100
    100-10
      100-10-10
    100-20
    100-30
  200
    200-10
    200-20
    200-30
  300
    300-10
    300-20
Diet
  100-10
    100-10-11
  200-10
  300-10
Bottle
  200-10
  300-20
```

```
@EQUAL("100-10", "Product")
```

Returns the members [Diet].[100-10] and [100].[100-10].

```
@EQUAL("100-10", "Diet")
```

Returns the member [Diet].[100-10].

See Also

- [@BETWEEN](#)
- [@EXPAND](#)
- [@LIKE](#)
- [@MBRCOMPARE](#)
- [@MBRPARENT](#)
- [@NOTEQUAL](#)

@EXP

Returns the exponent of a specified expression; that is, the value of e (the base of natural logarithms) raised to the power of the specified expression.

Syntax

@EXP (*expression*)

Parameter Description

expression Single member specification, variable name, function, or other numeric expression. If less than -700 or greater than 700, Essbase returns #MISSING.

Example

The following example is based on a variation of Sample Basic:

```
Index = @EXP("Variance %"/100);
```

This example produces the following result:

	East	West	South	Central
Variance %	10.7	10.9	3.6	3.6
Index	1.11293	1.11516	1.03666	1.03666

See Also

- [@LN](#)

@EXPAND

Expands a member search by calling a member set function for each member in a member list.

The members returned by the @EXPAND function are added to the existing member set.

Duplicate members are not removed from the member set.

This function can be used on unique and duplicate-name outlines.

Syntax

```
@EXPAND (mbrSetFunction, mbrList[, genLevNum][, LAYERONLY | ALL][, topMbrinHierarchy])
```

Parameter

Description

mbrSetFunction One of the following member set functions, which return a list of members:

- @ANCESTORS
- @IANCESTORS
- @CHILDREN
- @ICHILDREN
- @DESCENDANTS
- @IDESCENDANTS
- @EQUAL
- @MBRPARENT
- @SIBLINGS
- @ISIBLINGS

mbrList A comma-delimited list of members grouped together using @LIST or a member set function (such as @DESCENDANTS) that returns a list of members.

Parameter	Description
genLevNum	<p>Optional: This argument applies only if you specify @ANCESTORS, @IANCESTORS, @DESCENDANTS, or @IDESCENDANTS for <code>mbrSetFunction</code>.</p> <p>The integer value that defines the absolute generation or level number up to which to select members. A positive integer defines a generation number. A value of 0 or a negative integer defines a level number.</p>
LAYERONLY	<p>Optional: This argument applies only if you specify @ANCESTORS, @IANCESTORS, @DESCENDANTS, or @IDESCENDANTS for <code>mbrSetFunction</code>.</p> <p>Returns only those members at the specified generation or level (<i>genLevNum</i>) that match the selection criteria.</p> <p>If you specify this argument, you must specify <i>genLevNum</i>.</p>
ALL	<p>Optional: This argument applies only if you specify @ANCESTORS, @IANCESTORS, @DESCENDANTS, or @IDESCENDANTS for <code>mbrSetFunction</code>.</p> <p>Returns all of the members that match the member selection criteria, starting with the specified top member (<i>topMbrinHierarchy</i>).</p> <p>If you specify this argument, you must specify <i>topMbrinHierarchy</i>.</p>
topMbrinHierarchy	<p>Optional: This argument applies only if you specify @EQUAL for <code>mbrSetFunction</code>.</p> <p>A fully qualified member name on which to base the member search. The specified member and its aliases, and all of its descendants, are included in the search.</p> <p>If you specify @EQUAL for <code>mbrSetFunction</code>, and you do not specify <i>topMbrinHierarchy</i>, Essbase searches the entire outline.</p>

Example

The following examples are based on the following duplicate-name outline:

```

Product
  100
    100-10
      100-10-10
    100-20
    100-30
  200
    200-10
    200-20
    200-30
  300
    300-10
    300-20
Diet
  100-10
    100-10-11
  200-10
  300-10
Bottle
  200-10
  300-20

```

```
@EXPAND("@DESC", @LIST("Product"), -1, LAYERONLY)
```

Returns all of the members under the Product dimension that are at level 1, which are [100]. [100-10], [Product].[200], [Product].[300], [Diet].[100-10], and [Product].[Bottle].

```
@EXPAND("@EQUAL", @EXPAND("@CHILDREN", @LIST("[product].[100]", "[product].[200]")), , , "Product")
```

Essbase first executes the inner @EXPAND function—@EXPAND("@CHILDREN", @LIST("[product].[100]", "[product].[200]"))—which expands the member list to include all of the children of members 100 and 200 (a total of six members). Then Essbase executes the outer @EXPAND function, which searches the Product hierarchy for a match with any of the six members.

See Also

- [@BETWEEN](#)
- [@EQUAL](#)
- [@NOTEQUAL](#)
- [@LIKE](#)
- [@MBRCOMPARE](#)
- [@MBRPARENT](#)

@FACTORIAL

Returns the factorial of *expression*. The factorial of a number is equal to $1*2*3*...*$ number.

Syntax

```
@FACTORIAL (expression)
```

Parameter Description

expression Single member specification or numeric expression.

Notes

- *expression* can be no larger than 189. If *expression* is larger than 189, Essbase returns #MISSING.
- If *expression* is negative, Essbase returns #MISSING.

Example

```
@FACTORIAL (1)      1
@FACTORIAL (5)      120
```

See Also

- [@POWER](#)

@FORMATDATE

Returns a formatted date-string.

Syntax

@FormatDate(*date*, *date_format_string*)

Parameter	Description
<date>	<p>A number representing the input date between January 1, 1970 and Dec 31, 2037. The number is the number of seconds elapsed since midnight, January 1, 1970. To retrieve this number, use any of the following functions: @TODAY, @TODATEEX, @DATEROLL.</p> <p>Date-time attribute properties of a member can also be used to retrieve this number. For example, @AttributeVal("Intro Date"); returns the product introduction date for the current product in context.</p>
date_format_string	<p>One of the following literal strings (excluding ordered-list numbers and parenthetical examples) indicating a supported date format.</p> <ol style="list-style-type: none">1. "mon dđ yyyy" (Example: mon = Aug)2. "Month dđ yyyy" (Example: Month = August)3. "mm/dđ/yy"4. "mm/dđ/yyyy"5. "yy.mm.dđ"6. "dđ/mm/yy"7. "dđ.mm.yy"8. "dđ-mm-yy"9. "dđ Month yy"10. "dđ mon yy"11. "Month dđ, yy"12. "mon dđ, yy"13. "mm-dđ-yy"14. "yy/mm/dđ"15. "yymmdd"16. "dđ Month yyyy"17. "dđ mon yyyy"18. "yyyy-mm-dđ"19. "yyyy/mm/dđ"20. Long format (Example: WeekDay, Mon dđ, yyyy)21. Short format (Example: m/d/yy)

Notes

- Using an invalid input date returns an error.
- Using extra whitespace not included in the internal format strings returns an error.
- This function interprets years in the range 1970 to 2029 for yy format. Therefore, if the function is invoked using a date format mm/dd/yy for June 20, 2006, the returned date string is "06/20/06".

Example

Assume the outline has a date type member MyDate1.

```
Profit (If(@ToDateEx("YYYY-mm-dd", @FormatDate(@Today(), "YYYY-mm-dd")) == MyDate1 )
    Profit=99;
Endif;)
```

See Also

- [@DATEDIFF](#)
- [@DATEPART](#)
- [@DATEROLL](#)
- [@TODATEEX](#)
- [@TODAY](#)

@GEN

Returns the generation number of the specified member.

Syntax

@GEN (*mbrName*)

Parameter Description

mbrName Any valid single member name or member combination, or a function that returns a single member or member combination.

Example

In the Sample Basic database:

```
@GEN(Year)
```

Returns 1.

```
@GEN(Qtr3)
```

Returns 2.

See Also

- [@CURGEN](#)
- [@LEV](#)

@GENMBRS

Returns all members with the specified generation number or generation name in the specified dimension.

Syntax

@GENMBRS (*dimName*, *genName/genNum*)

Parameter	Description
dimName	A single dimension name specification.
genName genNum	Generation name or generation number from <i>dimName</i> . A positive integer defines a generation number.

Notes

- If you specify a name for the *genName* parameter, Essbase looks for a generation with that name in the specified dimension.
- If you specify a number for the *genName* parameter (for example, 2), Essbase first looks for a generation with a number string name. If no generation name exists with that numeric name, Essbase checks to see if the parameter is a valid generation number. Check the application event log after running the calculation to make sure that the correct members were calculated.
- Generation 0 is not a valid generation number. In Essbase, generations begin numbering at 1.
- If you specify a temporary variable for the *genName* parameter, Essbase does not recognize the value of the variable. It looks in the outline for a generation name with the same name as the temporary variable.
- For more information about generations and defining generation names, see the *Oracle Essbase Database Administrator's Guide*.
- Essbase sorts the generated list of members in ascending order. Using Sample Basic as an example, if you specify `@GENMBRS (Product, 2)`, Essbase returns 100, 200, 300, 400, Diet (in that order). This order is important to consider when you use the `@GENMBRS` member set function with certain forecasting and statistical functions.

Example

In the Sample Basic database:

```
@GENMBRS (Year, Month)
@GENMBRS (Year, 3)
```

both return the following members since generation 3 of the Year dimension is named Month:

Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, and Dec (in that order).

The following example restricts the calculation to members with the combination Budget and one of the members of the Market dimension with a generation name of State.

```
FIX (Budget, @GENMBRS (Market, State))
CALC DIM (Year, Measures);
ENDFIX
```

See Also

- [@LEVMBRS](#)

@GROWTH

Calculates a series of values that represent a linear growth of the first nonzero value encountered in *principalMbr* across the specified *rangeList*. Growth is calculated by multiplying the growth rate in *rateMbrConst* by the original *principalMbr*. This value is then added to the previous time period's result, yielding the new value.

Syntax

```
@GROWTH (principalMbr, rateMbrConst [, rangeList])
```

Parameter Description

principalMbr	Single member specification that represents the initial value of the value to grow. The first nonzero value encountered is the initial value. Other <i>principalMbr</i> values after the first are ignored.
rateMbrConst	Single member specification, variable name, or numeric expression providing a constant value that represents the decimal growth rate to be applied (for example, 10% = .1).
rangeList	Optional. A valid member name, a comma-delimited list of member names, member set functions, and range functions. If <i>rangeList</i> is not specified, Essbase uses the level 0 members from the dimension tagged as Time.

Notes

Financial functions never return a value; rather, they calculate a series of values internally based on the range specified.

Example

The following example calculates the growth of Principal Amount, using the rate found in Growth Rate for each fiscal year. The results are placed in Growth Amount.

```
"Growth Amount"=@GROWTH("Principal Amount", "Growth Rate", FY1998:FY2003);
```

This example produces the following report:

	FY1998	FY1999	FY2000	FY2001	FY2002	FY2003
Principal Amount	1,000	0	2,000	0	0	0
Growth Amount	1,050	1,120	1,200	1,280	1,380	1,480
Growth Rate	0.05	0.07	0.08	0.08	0.1	0.1

See Also

- [@COMPOUNDGROWTH](#)
- [@DECLINE](#)

@IALLANCESTORS

Returns the specified member and all the ancestors of that member, including ancestors of any occurrences of the specified member as a shared member. You can use this member set function as a parameter of another function, where that parameter is a list of members.

Syntax

@IALLANCESTORS (*mbrName*)

Parameter Description

mbrName A valid single member name or member combination, or a function that returns a single member or member combination.

Notes

Essbase sorts the generated list of members in ascending order of the member number in the outline. Using Sample Basic as an example, if you specify 100-20 for *mbrName*, Essbase returns 100-20, 100, Diet, Product (in that order). However, the order in which shared ancestors are returned is not guaranteed. This order is important to consider when you use the @IALLANCESTORS member set function with certain forecasting and statistical functions.

Example

The following example is based on the Sample Basic database. Sample Basic has a shared level of diet drinks, which includes 100-20 (Diet Cola). So 100-20 (Diet Cola) is a descendant of 100 (Colas) and is a shared member descendant of Diet:

```
100
    100-10
    100-20
    ...
Diet
    100-20 (Shared Member)
    ...
```

The following calculation script increases by 5% the Budget Sales values of 100-20 and all its ancestors, including Diet:

```
FIX(Budget,@IALLANCESTORS("100-20"))
Sales = Sales * 1.05;
ENDFIX
```

This example produces the following report. This report shows that the Budget->Sales values for 100-20, 100, Diet, and Product (100-20 and its ancestors) have been increased by 5%. The original values were 2610, 8980, 8260, and 28480, respectively.

		Jan	
		Actual	Budget
		Sales	Sales
		=====	=====
Market	100-10	4860	5200
	100-20	2372	2740.5 *
	100-30	1082	1170
	100	8314	9429 *
	100-20	2372	2610
	200-20	3122	3090
	300-30	2960	2560
	Diet	8454	8673 *
	Product	31538	30954 *

See Also

- [@ALLANCESTORS](#)
- [@IANCESTORS](#)
- [@LANCESTORS](#)
- [@ILANCESTORS](#)

@IANCESTORS

Returns the specified member and either all ancestors of the member or all ancestors up to the specified generation or level.

Essbase sorts the generated list of members—starting with the specified member, followed by the nearest ancestor of the member, followed by the next nearest ancestor of the member, and so on. In the Sample.Basic database, if you specify `@IANCESTORS(200-30)`, Essbase returns 200-30, 200, Product (in that order). When using the `@IANCESTORS` function with certain forecasting and statistical functions, you must consider order.

You can use the `@IANCESTORS` function as a parameter of another function, where the function requires a list of members.

Syntax

```
@IANCESTORS (mbrName [, genLevNum | genLevName])
```

Parameter	Description
-----------	-------------

<code>mbrName</code>	Valid member name or member-name combination or a function that returns one member or member combination.
----------------------	-----------------------------------------------------------------------------------------------------------

<code>genLevNum</code>	Optional. The integer value that defines the absolute generation or level number up to which to select members. A positive integer defines a generation number. A value of 0 or a negative integer defines a level number.
------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<code>genLevName</code>	Optional. The level or generation name up to which to select members.
-------------------------	-----------------------------------------------------------------------

Example

All examples are from the Sample.Basic database.

```
@IANCESTORS("New York")
```

Returns New York, East, Market (in that order).

```
@IANCESTORS(Qtr4)
```

Returns Qtr4, Year (in that order).

```
@IANCESTORS(Sales, -2)
```

Returns Sales, Margin, Profit (in that order). Members higher than level 2 are not returned.

```
@IANCESTORS("100-10", 1)
```

Returns 100-10, 100, Product (in that order). All ancestors are returned up to generation 1.

See Also

- [@ANCESTORS](#)
- [@IALLANCESTORS](#)
- [@LANCESTORS](#)
- [@ILANCESTORS](#)

@ICHILDREN

Returns the specified member and all of its children. This member set function can be used as a parameter of another function, where that parameter is a list of members.

Syntax

`@ICHILDREN (mbrName)`

Parameter Description

mbrName Any valid single member name or member combination, or a function that returns a single member or member combination.

Notes

Essbase sorts the generated list of members starting with the specified member, followed by its children in ascending order. Using Sample Basic as an example, if you specify 100 for *mbrName*, Essbase returns 100, 100-10, 100-20, 100-30 (in that order). This order is important to consider when you use the @ICHILDREN member set function with certain forecasting and statistical functions.

Example

In the Sample Basic database:

```
@ICHILDREN (Market)
```

Returns Market, East, West, South, and Central (in that order).

```
@ICHILDREN (Margin)
```

Returns Margin, Sales, and COGS (in that order).

See Also

- [@CHILDREN](#)

@IDESCENDANTS

Returns the specified member and either all descendants of the member or all descendants down to the specified generation or level.

Essbase sorts the generated list of members—starting with the specified member, followed by the nearest descendant of the member, followed by the next nearest descendant of the member, and so on. In the Sample.Basic database, if you specify @IDESCENDANTS (100), Essbase returns

100, 100-10, 100-20, 100-30 (in that order). When using the @IDESCENDANTS function with certain forecasting and statistical functions, you must consider order.

You can use the @IDESCENDANTS function as a parameter of another function, where the function requires a list of members.

Syntax

```
@IDESCENDANTS (mbrName[, genLevNum | genLevName])
```

Parameter	Description
-----------	-------------

mbrName	Valid member name or member-name combination or a function that returns one member or member combination.
---------	-----------------------------------------------------------------------------------------------------------

genLevNum	Optional. The integer value that defines the absolute generation or level number up to which to select members. A positive integer defines a generation number. A value of 0 or a negative integer defines a level number.
-----------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

genLevName	Optional. The level or generation name up to which to select members.
------------	-----------------------------------------------------------------------

Example

All examples are from the Sample.Basic database.

```
@IDESCENDANTS (East)
```

Returns East, New York, Massachusetts, Florida, Connecticut, and New Hampshire (in that order).

```
@IDESCENDANTS (Profit)
```

Returns Profit, Margin, Sales, COGS, Total Expenses, Marketing, Payroll, and Misc (in that order).

```
@IDESCENDANTS (Market, 2)
```

Returns Market, East, West, South, and Central (in that order).

```
@IDESCENDANTS (South, -1)
```

Returns South.

See Also

- [@DESCENDANTS](#)
- [@IRDESCENDANTS](#)
- [@RDESCENDANTS](#)
- [@LDESCENDANTS](#)
- [@ILDESCENDANTS](#)
- [@ISDESC](#)
- [@ANCESTORS](#)
- [@CHILDREN](#)
- [@SIBLINGS](#)

@ILANCESTORS

Returns the members of the specified member list and either all ancestors of the members or all ancestors up to the specified generation or level.

You can use the @ILANCESTORS function as a parameter of another function, where the function requires a list of members.

Syntax

```
@ILANCESTORS ((memberSetFunction) [, genLevNum])
```

Parameter	Description
memberSetFunction	<p>A member set function that returns a list of members.</p> <p>How the @ILANCESTORS function is used determines which member set functions are allowed. Follow these guidelines:</p> <ul style="list-style-type: none">• If the @ILANCESTORS function is used alone (not within a FIX statement), you must use the @LIST function and specify member names. For example: <code>@LIST (mbr1, mbr2, ...)</code>• If the @ILANCESTORS function is used within a FIX statement, you can use member set functions such as @UDA and @ATTRIBUTE. For example: <code>@UDA (dimName, uda)</code> <code>@ATTRIBUTE (attMbrName)</code> <p>In this case, you can choose whether to use the @LIST function. For example, both of the following statements are valid, and the statements return the same results.</p> <p>Example using only @ATTRIBUTE:</p> <pre>FIX (@ILANCESTORS (@ATTRIBUTE (Caffeinated_True), @ATTRIBUTE (Ounces_12), "200-40")) ... ENDFIX;</pre> <p>Example using @LIST and @ATTRIBUTE:</p> <pre>FIX (@ILANCESTORS (@LIST (@ATTRIBUTE (Caffeinated_True), @ATTRIBUTE (Ounces_12), "200-40")) ... ENDFIX;</pre>
genLevNum	<p>Optional. The integer value that defines the absolute generation or level number up to which to select members. A positive integer defines a generation number. A value of 0 or a negative integer defines a level number.</p>

Caution! All members of the specified member list must be from the same dimension.

Example

All examples are from the Sample.Basic database.

```
@ILANCESTORS (@LIST ("100-10", "200-20"))
```


Returns 100-10 (a specified member); 100 and Product (the ancestors of 100-10); 200-20 (a specified member); and 200 (the ancestor of 200-20). The result does not contain duplicate members.

```
@ILANCESTORS (@LIST ("100", "100-10"))
```

Returns 100 and 100-10 (the specified members); and Product (the ancestor of 100 and 100-10). The result does not contain duplicate members.

```
@ILANCESTORS (@LIST ("100", "Product", "200"))
```

Returns 100, Product, and 200 (the specified members). The result does not contain duplicate members.

```
FIX (@ILANCESTORS (@UDA (Market, "New Market"), 2)  
...  
ENDFIX;
```

Returns Nevada (a member that is assigned the New Market UDA) and West (the ancestor to generation 2 for Nevada); Louisiana (a member that is assigned the New Market UDA) and South (the ancestor to generation 2 for Louisiana); and Colorado (a member that is assigned the New Market UDA) and Central (the ancestor to generation 2 for Colorado).

```
FIX (@ILANCESTORS (@ATTRIBUTE (Caffeinated_True), @ATTRIBUTE (Ounces_12), "200-40")  
...  
ENDFIX;
```

Returns 100-10, 100-20, 200-10, and 300-30 (caffeinated, 12-ounce drinks); and 200-40 (the specified member), and 100, 200, 300, and Product (the ancestors of the members).

See Also

- [@LANCESTORS](#)
- [@ANCESTORS](#)
- [@IANCESTORS](#)

@ILDESCENDANTS

Returns the members of the specified member list and either all descendants of the members or all descendents down to the specified generation or level.

You can use the @ILDESCENDANTS function as a parameter of another function, where the function requires a list of members.

Syntax

```
@ILDESCENDANTS ( (memberSetFunction) [, genLevNum] )
```

Parameter	Description
memberSetFunction	<p>A member set function that returns a list of members.</p> <p>How the @ILDESCENDANTS function is used determines which member set functions are allowed. Follow these guidelines:</p> <ul style="list-style-type: none"> • If the @ILDESCENDANTS function is used alone (not within a FIX statement), you must use the @LIST function and specify member names. For example: <pre>@LIST (mbr1, mbr2, . . .)</pre> • If the @ILDESCENDANTS function is used within a FIX statement, you can use member set functions such as @UDA and @ATTRIBUTE. For example: <pre>@UDA (dimName, uda) @ATTRIBUTE (attMbrName)</pre> <p>In this case, you can choose whether to use the @LIST function. For example, both of the following statements are valid, and the statements return the same results.</p> <p>Example using only @ATTRIBUTE:</p> <pre>FIX (@ILDESCENDANTS (@ATTRIBUTE (Caffeinated_True), @ATTRIBUTE (Ounces _12), "200-40")) . . . ENDFIX;</pre> <p>Example using @LIST and @ATTRIBUTE:</p> <pre>FIX (@ILDESCENDANTS (@LIST (@ATTRIBUTE (Caffeinated_True), @ATTRIBUTE (Ounces_12), "200-40"))) . . . ENDFIX;</pre>

Caution! All members of the specified member list must be from the same dimension.

genLevNum Optional. The integer value that defines the absolute generation or level number up to which to select members. A positive integer defines a generation number. A value of 0 or a negative integer defines a level number.

Example

All examples are from the Sample.Basic database.

```
@ILDESCENDANTS (@LIST ("100", "200", "300"))
```

Returns 100 (a specified member); 100-10, 100-20, 100-30 (the descendants of 100); 200 (a specified member); and 200-10, 200-20, 200-30, and 200-40 (the descendants of 200); 300 (a specified member); and 300-10, 300-20, 300-30 (the descendants of 300).

```
@ILDESCENDANTS (@LIST ("Market"), -1)
```

Returns Market (the specified member); and East, West, South, and Central (the descendants of Market to level 1).

```
FIX
  (@ILDESCENDANTS (@UDA (Market, "Major Market")))
```

```
...  
ENDFIX;
```

Returns East (a specified member); New York, Massachusetts, Florida, Connecticut, and New Hampshire (the descendants of East); Central (a specified member); Illinois, Ohio, Wisconsin, Missouri, Iowa, and Colorado (the descendants of Central); California and Texas (specified members, which do not have descendants).

```
FIX  
(@ILDESCENDANTS(@ATTRIBUTE(Caffeinated_True)@ATTRIBUTE(Ounces_12), "200-40")  
...  
ENDFIX;
```

Returns 100-10, 100-20, 200-10, 300-30 (caffeinated, 12-ounce drinks); and 200-40 (a specified member). None of these members have descendants.

See Also

- [@LDESCENDANTS](#)
- [@IDESCENDANTS](#)
- [@RDESCENDANTS](#)
- [@IRDESCENDANTS](#)
- [@ISDESC](#)
- [@ANCESTORS](#)
- [@LANCESTORS](#)
- [@ILANCESTORS](#)
- [@CHILDREN](#)
- [@SIBLINGS](#)
- [@SHIFTSIBLING](#)

@ILSIBLINGS

Returns the specified member and its left siblings.

Syntax

```
@ILSIBLINGS (mbrName)
```

Parameter Description

mbrName Any valid single member name or member combination, or a function that returns a single member or member combination.

Notes

This function returns the specified member and all of the left siblings of the member. Left siblings are children that share the same parent as the member and that precede the member in the database outline.

This member set function can be used as a parameter of another function, where that parameter is a list of members.

Essbase sorts the generated list of members starting with the left siblings of the member (that is, siblings appearing above the member in the database outline) in ascending order. Using Sample Basic as an example, if you specify 200-30 for *mbrName*, Essbase returns 200-10, 200-20, 200-30 (in that order). This order is important to consider when you use the @ILSIBLINGS member set function with certain forecasting and statistical functions.

Example

In the Sample Basic database:

```
@ILSIBLINGS(Florida)
```

Returns New York, Massachusetts, and Florida (in that order). New York and Massachusetts appear above Florida in the Sample Basic outline.

```
@ILSIBLINGS(Qtr3)
```

Returns Qtr1, Qtr2, and Qtr3 (in that order). Qtr1 and Qtr2 appear above Qtr3 in the Sample Basic outline.

See Also

- [@LSIBLINGS](#)

@INT

Returns the next lowest integer value of *expression*.

Syntax

```
@INT (expression)
```

Parameter Description

expression Member specification or mathematical expression that generates a numeric value.

Example

The following example is based on the Sample Basic database. Assume that the Profit % member is not tagged as Dynamic Calc.

The following formula rounds the values for West down to the nearest integer.

```
West=@INT (@SUM (@CHILDREN (West) ) ) ;
```

This example produces the following report:

	Profit %		
	Cola		Actual
	Jan	Feb	Mar
	===	===	===
California	38.64	37.98	38.37
Oregon	17.50	16.13	16.11
Washington	29.23	30.90	32.00
Utah	23.08	23.08	20.97
Nevada	-3.95	-6.76	-5.33
West	104	101	102

See Also

- [@ABS](#)
- [@REMAINDER](#)
- [@ROUND](#)
- [@TRUNCATE](#)

@INTEREST

Calculates the simple interest in *balanceMbr* at the rate specified by *creditrteMbrConst* if the value specified by *balanceMbr* is positive, or at the rate specified by *borrowrateMbrConst* if *balanceMbr* is negative. The interest is calculated for each time period specified by *rangeList*.

Syntax

```
@INTEREST (balanceMbr, creditrteMbrConst, borrowrateMbrConst  
          [, rangeList])
```

Parameter	Description
<i>balanceMbr</i>	Single member specification representing the balance at the time the interest is calculated.
<i>creditrteMbrConst</i>	Single member specification, variable name, or numeric expression providing a constant value. The value must be a decimal number that corresponds to a percentage. The value represents the per-period interest rate.
<i>borrowrateMbrConst</i>	Single member specification, variable name, or numeric expression providing a constant value. The value must be a decimal number corresponding to a percentage value. The value represents the per-period interest rate.
<i>rangeList</i>	Optional. A valid member name, a comma-delimited list of member names, member set functions, and range functions from the dimension tagged as Time. If <i>rangeList</i> is not specified, Essbase uses the level 0 members from the dimension tagged as Time.

Notes

Financial functions never return a value; rather, they calculate a series of values internally based on the range specified.

Example

This example calculates the interest for Balance, using Credit Rate for positive balances and using Borrow Rate for negative balances. The results are placed in Interest Amount for each fiscal year.

```
"Interest Amount" = @INTEREST(Balance, "Credit Rate", "Borrow Rate",  
FY1998:FY2001, FY2002, FY2003);
```

This example produces the following report:

	FY1998	FY1999	FY2000	FY2001	FY2002	FY2003
	=====	=====	=====	=====	=====	=====
Balance	2000.00	3000.00	-1000.00	3000.00	9000.00	-6000.00
Credit Rate	0.065	0.065	0.065	0.065	0.065	0.065

Borrow Rate	0.1125	0.1125	0.1125	0.1125	0.1125	0.1125
Interest Amount	130.00	195.00	-112.50	195.00	585.00	-675.00

See Also

- [@COMPOUND](#)

@IRDESCENDANTS

Returns the specified member and all its descendants, or all descendants down to a specified generation or level, including descendants of any occurrences of the specified member as a shared member.

You can use this member set function as a parameter of another function, where that parameter is a list of members. In the absence of shared members, [@IRDESCENDANTS](#) and [@IDESCENDANTS](#) have identical behavior.

Syntax

`@IRDESCENDANTS (mbrName[, genLevNum | genLevName])`

Parameter Description

<code>mbrName</code>	Any valid single member name or member combination, or a function that returns a single member or member combination
<code>genLevNum</code>	Optional. An integer value that defines the absolute generation or level number down to which to select the members. A positive integer defines a generation number. A value of 0 or a negative integer defines a level number.
<code>genLevName</code>	Optional. Level name or generation name down to which to select the members.

Notes

- The order of members in the result list is important to consider when you use the [@IRDESCENDANTS](#) member set function with certain forecasting and statistical functions. Essbase generates the list of members in the following sequence: If a shared member is encountered, the above steps are repeated on the member being shared.
 1. The specified member
 2. The nearest descendant of the member
 3. The next nearest descendant of the member, and so on
- You can use [@RDESCENDANTS](#) to exclude the specified member and include descendants of shared members.
- You can use [@IDESCENDANTS](#) to include the specified member and exclude descendants of shared members.
- You can use [@DESCENDANTS](#) to exclude the specified member and descendants of shared members.

Example

Example 1

Assume a variation of the Sample Basic database such that the Product dimension includes the following members:

```
Product
  100
    100-10
    100-20
    100-30
  200
    200-10
    200-20
    200-30
    200-40
Diet
  100 (Shared Member)
  200 (Shared Member)
```

Diet has two children "100" and "200" instead of "100-10", "200-20" and "300-30". The members "100" and "200" are shared members.

```
@IRDESCENDANTS (Diet)
```

Returns the members: Diet, 100, 100-10, 100-20, 100-30, 200, 200-10, 200-20, 200-30, 200-40 (in that order).

Example 2

```
@IRDESCENDANTS (East)
```

Returns East, New York, Massachusetts, Florida, Connecticut, and New Hampshire (in that order) and is exactly the same as @IDESCENDANTS(East).

See Also

- [@RDESCENDANTS](#)
- [@IDESCENDANTS](#)
- [@DESCENDANTS](#)
- [@ISDESC](#)
- [@ICHLDRN](#)
- [@ISIBLINGS](#)
- [@IANCESTORS](#)

@IRR

Calculates the Internal Rate of Return on a cash flow that must contain at least one investment (negative) and one income (positive) value.

Syntax

```
@IRR (cashflowMbr, discountFlag[, rangeList])
```

Parameter	Description
-----------	-------------

cashflowMbr	Single member specification.
-------------	------------------------------

Parameter Description

discountFlag Member specification, variable name, or numeric expression providing a constant value of either 1 or 0. *discountFlag* indicates whether the function should discount from the first period. 1 means do not discount from the first period.

rangeList Optional. A valid member name, a comma-delimited list of member names, member set functions, and range functions. If *rangeList* is not specified, Essbase uses the level 0 members from the dimension tagged as Time.

Notes

- Financial functions never return a value; rather, they calculate a series of values internally based on the range specified.
- Essbase returns #MISSING from calculator function @IRR if all cash flows are zero.
- @IRR provides an initial guess of 0.07. This cannot be changed, in contrast to similar functions in Excel. Because results depend in part on the initial guess, any difference in the initial guess may result in a different result. Even if both Excel and Essbase start with the same initial guess, results may differ. This is because there may be more than one solution to an equation, and the algorithm stops looking when it finds a valid solution. Which solution is found first may differ based on the algorithm. Although leading or trailing zeros do not matter in a mathematical context, the algorithm may behave differently and find a different root because of the presence of leading or trailing zeros. If you need identical solutions regardless of the presence of leading or trailing zeros, you may wish to create a custom-defined function to handle these issues.

Example

This example calculates the Internal Rate of Return (Return) on a cash flow (Cash).

```
Return = @IRR(Cash,0,FY1998:FY2000,FY2001:FY2003);
```

This example produces the following report:

	FY1998	FY1999	FY2000	FY2001	FY2002	FY2003
	=====	=====	=====	=====	=====	=====
Cash	(1,000)	500	600	500	#MISSING	#MISSING
Rate	0	0	0	0	#MISSING	#MISSING
Return	0	0	0	0	0	0

@IRSIBLINGS

Returns the specified member and its right siblings.

Syntax

```
@IRSIBLINGS (mbrName)
```

Parameter Description

mbrName Any valid single member name or member combination, or a function that returns a single member or member combination.

Notes

This function returns the specified member and all of the right siblings of the specified member. Right siblings are children that share the same parent as the member and that follow the member in the database outline.

This member set function can be used as a parameter of another function, where that parameter is a list of members.

Essbase sorts the generated list of members starting with the specified member, followed by the right siblings of the member (that is, siblings appearing below the member in the database outline) in ascending order. Using Sample Basic as an example, if you specify 200-20 for *mbrName*, Essbase returns 200-20, 200-30, 200-40 (in that order). This order is important to consider when you use the @IRSIBLINGS member set function with certain forecasting and statistical functions.

Example

In the Sample Basic database:

```
@IRSIBLINGS(Florida)
```

Returns Florida, Connecticut, and New Hampshire (in that order). Connecticut and New Hampshire appear below Florida in the Sample Basic outline.

```
@IRSIBLINGS(Qtr3)
```

Returns Qtr3 and Qtr4 (in that order). Qtr4 appears below Qtr3 in the Sample Basic outline.

See Also

- [@RSIBLINGS](#)

@ISACCTYPE

Returns TRUE if the current member has the associated accounts tag.

Syntax

```
@ISACCTYPE (tag)
```

Parameter Description

tag	Valid account tag defined in the current database. Any of these values may be used: First, Last, Average, Expense, and Twopass. To ensure that the tag is resolved as a string rather than a member name, it is recommended to enclose it in quotation marks.
-----	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Example

The following example is based on the Sample Basic database. For members with the Expense accounts tag, the formula uses the @ABS function to calculate Budget as the absolute value of Budget.

```
IF (@ISACCTYPE("Expense"))  
    Budget = @ABS(Budget);  
ENDIF;
```

@ISANCEST

Returns TRUE if the current member is an ancestor of the specified member. This function excludes the specified member.

Syntax

```
@ISANCEST (mbrName)
```

Parameter	Description
-----------	-------------

<i>mbrName</i>	Any valid single member name or member combination, or a function that returns a single member or member combination.
----------------	-----------------------------------------------------------------------------------------------------------------------

Example

In the Sample Basic database:

```
@ISANCEST (California)
```

Returns TRUE for Market, West

```
@ISANCEST (West)
```

Returns FALSE for California, West, East

See Also

- [@ISANCEST](#)

@ISATTRIBUTE

Returns TRUE if the current member under calculation matches the attribute or varying attribute name specified in *attmbrName*.

Syntax

```
@ISATTRIBUTE (attMbrName)
```

Parameter	Description
-----------	-------------

<i>attMbrName</i>	Single attribute member name or member combination.
-------------------	-----------------------------------------------------

Notes

- This function provides the same functionality as `@IsMbr (@Attribute(attMbrName))`, but is faster.
- You may have duplicate Boolean, date, and numeric attribute member names in your outline. For example, 12 can be the attribute value for the size (in ounces) of a product as well as the value for the number of packing units for a product. To distinguish duplicate member names, specify the full attribute member name (for example, `@ISATTRIBUTE (Ounces_12)`).

Example

Consider the following calculation script, based on the Sample Basic database:

```
/* To increase the marketing budget for markets with large populations */
Marketing (
  IF (@ISATTRIBUTE(Large))
    Marketing = Marketing * 1.1;
  ENDIF
);
```

See Also

- [@ISMBRWITHATTR](#)
- [SET SCAPERSPECTIVE](#)

@ISCHILD

Returns TRUE if the current member is a child of the specified member. This function excludes the specified member.

Syntax

@ISCHILD (*mbrName*)

Parameter Description

mbrName Any valid single member name or member combination, or a function that returns a single member or member combination.

Example

In the Sample Basic database:

```
@ISCHILD(East)
```

Returns TRUE for New York, Florida, Connecticut

```
@ISCHILD(Margin)
```

Returns FALSE for Measures, Profit, Margin

See Also

- [@ISCHILD](#)

@ISDESC

Returns TRUE if the current member is a descendant of the specified member. This function excludes the specified member.

Syntax

@ISDESC (*mbrName*)

Parameter Description

mbrName Any valid single member name or member combination, or a function that returns a single member or member combination.

Example

In the Sample Basic database:

```
@ISDESC (Market)
```

Returns TRUE for West, California, Oregon, Washington, Utah, Nevada

```
@ISDESC (Profit)
```

Returns FALSE for Measures, Profit, Profit %

@ISGEN

Returns TRUE if the current member of the specified dimension is in the specified generation.

Syntax

```
@ISGEN (dimName, genName | genNum)
```

Parameter

Description

dimName

The name of a dimension.

genName or genNum

Generation name specification, or a non-negative number that defines the number of a generation.

Example

In the Sample Basic database:

```
@ISGEN (Measures, 3)
```

Returns TRUE if the current member is Margin, Total Inventory, or Margin %, because these members are all in generation 3 of the Measures dimension.

```
@ISGEN (Market, 2)
```

Returns FALSE if the current member is New York or Market, because these members are not in generation 2 of the Market dimension.

See Also

- [@ISSAMEGEN](#)
- [@ISLEV](#)

@ISIANCEST

Returns TRUE if the current member is the specified member or an ancestor of the specified member. This function includes the specified member.

Syntax

@ISIANCEST (*mbrName*)

Parameter Description

mbrName Any valid single member name or member combination, or a function that returns a single member or member combination.

Example

In the Sample Basic database:

```
@ISIANCEST(California)
```

Returns TRUE for Market, West, and California. California is the specified member, and West and Market are ancestors of California.

```
@ISIANCEST(Qtr1)
```

Returns FALSE for Jan, Feb, Mar, Qtr2. None of these members is the specified member (Qtr1) or an ancestor of Qtr1.

See Also

- [@ISANCEST](#)

@ISIBLINGS

Returns the specified member and all siblings of that member. This member set function can be used as a parameter of another function, where that parameter is a list of members.

Syntax

@ISIBLINGS (*mbrName*)

Parameter Description

mbrName Any valid single member name or member combination, or a function that returns a single member or member combination.

Notes

Essbase sorts the generated list of members in ascending order. Using Sample Basic as an example, if you specify 200-30 for *mbrName*, Essbase returns 200-10, 200-20, 200-30, 200-40 (in that order). This order is important to consider when you use the @ISIBLINGS member set function with certain forecasting and statistical functions.

Example

In the Sample Basic database:

```
@ISIBLINGS(California)
```

returns California, Oregon, Washington, Utah, and Nevada (in that order), because these members are siblings of California.

@ISIBLINGS (Qtr2)

returns Qtr1, Qtr2, Qtr3, and Qtr4 (in that order), because these members are siblings of Qtr2.

See Also

- [@SIBLINGS](#)
- [@SHIFTSIBLING](#)
- [@NEXTSIBLING](#)
- [@PREVSIBLING](#)

@ISICHILD

Returns TRUE if the current member is the specified member or a child of the specified member.

Syntax

@ISICHILD (*mbrName*)

Parameter Description

mbrName Any valid single member name or member combination, or a function that returns a single member or member combination.

Example

In the Sample Basic database:

@ISICHILD (South)

Returns TRUE for Texas, Oklahoma, Louisiana, New Mexico, South

@ISICHILD (Profit)

Returns FALSE for Measures, Sales

See Also

- [@ISCHILD](#)

@ISIDESC

Returns TRUE if the current member is the specified member or a descendant of the specified member.

Syntax

@ISIDESC (*mbrName*)

Parameter Description

mbrName Any valid single member name or member combination, or a function that returns a single member or member combination.

Example

In the Sample Basic database:

```
@ISIDESC (South)
```

Returns TRUE for Texas, Oklahoma, Louisiana, New Mexico, South

```
@ISIDESC (West)
```

Returns FALSE for Market, East, South, and Central

See Also

- [@ISDESC](#)

@ISIPARENT

Returns TRUE if the current member is the specified member or the parent of the specified member.

Syntax

```
@ISIPARENT (mbrName)
```

Parameter Description

mbrName Any valid single member name or member combination, or a function that returns a single member or member combination.

Example

In the Sample Basic database:

```
@ISIPARENT (Qtr1)
```

Returns TRUE for Year, Qtr1.

```
@ISIPARENT (Margin)
```

Returns FALSE for Measures, Sales.

See Also

- [@ISPARENT](#)

@ISISIBLING

Returns TRUE if the current member is the specified member or a sibling of the specified member.

Syntax

```
@ISISIBLING (mbrName)
```

Parameter Description

mbrName Any valid single member name or member combination, or a function that returns a single member or member combination.

Example

In the Sample Basic database:

```
@ISISIBLING(Qtr2)
```

Returns TRUE for Qtr1, Qtr2, Qtr3, and Qtr4.

```
@ISISIBLING(Actual)
```

Returns FALSE for Scenario.

See Also

- [@ISSIBLING](#)

@ISLEV

Returns TRUE if the current member of the specified dimension is in the specified level.

Syntax

```
@ISLEV (dimName, levName | levNum)
```

Parameter Description

dimName Name of a dimension.

levName | levNum A level name or an integer value that defines the number of a level. A value of 0 or a negative integer defines a level number.

Example

In the Sample Basic database:

```
@ISLEV(Market, 0)
```

Returns TRUE if the current member of Market is New York, California, Texas, or Illinois.

```
@ISLEV(Year, 1)
```

Returns FALSE if the current member of Year is Jan, Feb, or Mar.

See Also

- [@ISSAMELEV](#)
- [@ISGEN](#)

@ISMBR

Returns TRUE if the current member matches any one of the specified members.

Syntax

`@ISMBR (mbrName | rangeList | mbrList)`

Parameter Description

mbrName	Any valid single member name or member combination, or a function that returns a single member or member combination.
rangeList	A valid member name, a comma-delimited list of member names, member set functions, and range functions.
mbrList	A comma-delimited list of members.

Notes

If a cross-dimensional (->) member is included, that term evaluates as TRUE only if all the components of the cross-dimensional member match the current member list.

If any term returns TRUE, the @ISMBR function returns TRUE.

Example

In the Sample Basic database:

```
@ISMBR("New York":"New Hampshire")
```

Returns TRUE for Florida.

```
@ISMBR(@CHILDREN(Qtr1))
```

Returns FALSE for Qtr2, Year.

@ISMBRWITHATTR

Returns TRUE if the current member belongs to the list of base members that are associated with an attribute that satisfies the conditions you specify.

Syntax

`@ISMBRWITHATTR (dimName, "operator", value)`

Parameter Description

dimName	Single varying attribute dimension name.
operator	Operator specification, which must be enclosed in quotation marks ("").
value	A value that, in combination with the operator, defines the condition that must be met. The <i>value</i> can be a varying attribute member specification, a constant, or a date-format function (that is, @TODATE).

Notes

- This function provides the same functionality as @IsMbr(@WithAttr()), but is faster.
- This function is a superset of the @ISATTRIBUTE function. The following two formulas return the same member set:

```
@ISATTRIBUTE(Bottle)
@ISMBRWITHATTR("Pkg Type", "=", Bottle)
```

However, the following formula can be performed only with @ISMBRWITHATTR (not with @ISATTRIBUTE) because you specify a condition:

```
@ISMBRWITHATTR(Ounces, ">", "16")
```

- If you specify a date attribute with the @ISMBRWITHATTR function, you must use the @TODATE function in the *string* parameter to convert the date string to a number. For more information, see the topic for the @TODATE function.
- The following operators are supported:

Operator	Description
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
= =	Equal to
<> or !=	Not equal to
IN	In

When using Boolean attributes with @ISMBRWITHATTR, use only the actual Boolean attribute member name, or use 1 (for True or Yes) or 0 (for False or No). You cannot use True/Yes and False/No interchangeably.

See Also

- [@WITHATTR](#)
- [@ISATTRIBUTE](#)
- [SET SCAPERSPECTIVE](#)
- [@ATTRIBUTE](#)
- [@ATTRIBUTEVAL](#)
- [@TODATE](#)

@ISPARENT

Returns TRUE if the current member is the parent of the specified member. This function excludes the specified member.

Syntax

```
@ISPARENT (mbrName)
```

Parameter Description

`mbrName` Any valid single member name or member combination, or a function that returns a single member or member combination.

Example

In the Sample Basic database:

```
@ISPARENT("New York")
```

Returns TRUE for East.

```
@ISPARENT(Profit)
```

Returns FALSE for Margin.

See Also

- [@ISIPARENT](#)

@ISSAMEGEN

Returns TRUE if the current member is the same generation as the specified member.

Syntax

```
@ISSAMEGEN (mbrName)
```

Parameter Description

`mbrName` Any valid single member name or member combination, or a function that returns a single member or member combination.

Example

In the Sample Basic database:

```
@ISSAMEGEN(West)
```

Returns TRUE for East.

```
@ISSAMEGEN(West)
```

Returns FALSE for California.

See Also

- [@ISGEN](#)
- [@GEN](#)
- [@ISSAMELEV](#)

@ISSAMELEV

Returns TRUE if the current member is the same level as the specified member.

Syntax

`@ISSAMELEV (mbrName)`

Parameter Description

`mbrName` Any valid single member name or member combination, or a function that returns a single member or member combination.

Example

In the Sample Basic database:

```
@ISSAMELEV(Sales)
```

Returns FALSE for Total Expenses.

```
@ISSAMELEV(Jan)
```

Returns TRUE for Apr, Jul, Oct.

See Also

- [@ISLEV](#)
- [@LEV](#)
- [@ISSAMEGEN](#)

@ISSIBLING

Returns TRUE if the current member is a sibling of the specified member. This function excludes the specified member.

Syntax

`@ISSIBLING (mbrName)`

Parameter Description

`mbrName` Any valid single member name or member combination, or a function that returns a single member or member combination.

Example

In the Sample Basic database:

```
@ISSIBLING("New York")
```

Returns TRUE for Florida, New Hampshire.

```
@ISSIBLING(Sales)
```

Returns FALSE for Margin.

See Also

- [@ISISIBLING](#)

@ISUDA

Returns TRUE if the specified user-defined attribute (UDA) exists for the current member of the specified dimension at the time of the calculation.

Syntax

```
@ISUDA (dimName, UDAStr)
```

Parameter Description

dimName Dimension name specification that contains the member you are checking.

UDAStr user-defined attribute (UDA) name string.

Notes

- Essbase checks to see if the UDA is defined for the current member of the specified dimension at calculation time. It returns TRUE if the UDA is defined, FALSE if not.
- For more information about UDAs, see the *Oracle Essbase Database Administrator's Guide*.

Example

The following example is based on the Sample Basic database. The Market dimension has members that indicate a geographic location. Some members represent major markets. The example below calculates the database and stores a budget amount for the upcoming year based on the actual amount from this year. A different sales growth rate is applied to major markets than to small markets.

```
FIX (Budget)
  Sales (IF(@ISUDA(Market, "Major Market"))
    Sales = Sales->Actual * 1.2;
  ELSE
    Sales = Sales->Actual * 1.1;
  ENDF;);
ENDFIX
```

The preceding example tests to see if the current member of Market has a UDA called "Major Market". If it does, the Budget -> Sales value is set to 120% of Actual -> Sales. If it does not, the Budget -> Sales value is set to 110% of Actual -> Sales.

See Also

- [@UDA](#)

@LANCESTORS

Returns all ancestors of the members in the specified member list or all ancestors up to a specified generation or level. This function excludes the specified members.

You can use the @LANCESTORS function as a parameter of another function, where the function requires a list of members.

Syntax

@LANCESTORS ((memberSetFunction) [, genLevNum])

Parameter	Description
-----------	-------------

memberSetFunction	A member set function that returns a list of members.
-------------------	-------------------------------------------------------

How the @LANCESTORS function is used determines which member set functions are allowed. Follow these guidelines:

- If the @LANCESTORS function is used alone (not within a FIX statement), you must use the @LIST function and specify member names. For example:

```
@LIST(mbr1, mbr2, . . .)
```

- If the @LANCESTORS function is used within a FIX statement, you can use member set functions such as @UDA and @ATTRIBUTE. For example:

```
@UDA(dimName, uda)
```

```
@ATTRIBUTE(attMbrName)
```

In this case, you can choose whether to use the @LIST function. For example, both of the following statements are valid, and the statements return the same results.

Example using only @ATTRIBUTE:

```
FIX(@LANCESTORS(@ATTRIBUTE(Caffeinated_True), @ATTRIBUTE(Ounces_12), "200-40"))
. . .
ENDFIX;
```

Example using @LIST and @ATTRIBUTE:

```
FIX(@LANCESTORS(@LIST(@ATTRIBUTE(Caffeinated_True), @ATTRIBUTE(Ounces_12), "200-40")))
. . .
ENDFIX;
```

Caution! All members of the specified member list must be from the same dimension.

genLevNum	Optional. The integer value that defines the absolute generation or level number up to which to select members. A positive integer defines a generation number. A value of 0 or a negative integer defines a level number.
-----------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Example

All examples are from the Sample.Basic database.

```
@LANCESTORS(@LIST("100-10", "200-20"), 2)
```

Returns 100 (the ancestor of 100-10); and 200 (the ancestor of 200-20). Excludes Product because it is at generation 1.

```
@LANCESTORS(@LIST("100", "100-10"))
```

Returns Product (the ancestor of 100); and 100 (the ancestor of 100-10). The result does not contain duplicate members.

```
@LANCESTORS(@LIST("100", "Product", "200"))
```

Returns Product (the ancestor of 100 and 200). The result does not contain duplicate members.

```
FIX(@LANCESTORS(@UDA(Market, "New Market")), 2)
...
ENDFIX;
```

Returns West, South, and Central (the ancestors, to generation 2, for the members in the Market dimension that are associated with the New Market attribute).

```
FIX(@LANCESTORS(@ATTRIBUTE(Caffeinated_True), @ATTRIBUTE(Ounces_12), "200-40"))
...
ENDFIX;
```

Returns 100, 200, 300, and Product (the ancestors of 100-10, 100-20, 200-10, 300-30—caffeinated, 12-ounce drinks, and 200-40).

See Also

- [@ILANCESTORS](#)
- [@ANCESTORS](#)
- [@IANCESTORS](#)

@LDESCENDANTS

Returns all descendants of the members in the specified member list or all descendants down to the specified generation or level. This function excludes the specified members.

You can use the @LDESCENDANTS function as a parameter of another function, where the function requires a list of members.

Syntax

```
@LDESCENDANTS ((memberSetFunction) [, genLevNum])
```

Parameter	Description
memberSetFunction	<p>A member set function that returns a list of members.</p> <p>How the @LDESCENDANTS function is used determines which member set functions are allowed. Follow these guidelines:</p> <ul style="list-style-type: none"> • If the @LDESCENDANTS function is used alone (not within a FIX statement), you must use the @LIST function and specify member names. For example: <pre>@LIST (mbr1, mbr2, ...)</pre> • If the @LDESCENDANTS function is used within a FIX statement, you can use member set functions such as @UDA and @ATTRIBUTE. For example: <pre>@UDA (dimName, uda) @ATTRIBUTE (attMbrName)</pre> <p>In this case, you can choose whether to use the @LIST function. For example, both of the following statements are valid, and the statements return the same results.</p> <p>Example using only @ATTRIBUTE:</p> <pre>FIX (@LDESCENDANTS (@ATTRIBUTE (Caffeinated_True) , @ATTRIBUTE (Ounces_12) , "200-40")) ... ENDFIX;</pre> <p>Example using @LIST and @ATTRIBUTE:</p> <pre>FIX (@LDESCENDANTS (@LIST (@ATTRIBUTE (Caffeinated_True) , @ATTRIBUTE (Ounces_12) , "200-40"))) ... ENDFIX;</pre>

Caution! All members of the specified member list must be from the same dimension.

genLevNum Optional. The integer value that defines the absolute generation or level number up to which to select members. A positive integer defines a generation number. A value of 0 or a negative integer defines a level number.

Example

All examples are from the Sample.Basic database.

```
@LDESCENDANTS (@LIST ("100" , "200" , "300" ) )
```

Returns 100-10, 100-20, 100-30 (the descendants of 100); 200-10, 200-20, 200-30, 200-40 (the descendants of 200); and 300-10, 300-20, 300-30 (the descendants of 300).

```
@LDESCENDANTS (@LIST ("Market" ) , -1)
```

Returns East, West, South, and Central (the descendants of the specified member Market to level 1).

```
FIX
  (@LDESCENDANTS (@UDA (Market, "Major Market" ) ) )
  ...
ENDFIX;
```


Returns New York, Massachusetts, Florida, Connecticut, and New Hampshire (the descendants of the specified member East); and Illinois, Ohio, Wisconsin, Missouri, Iowa, and Colorado (the descendants of the specified member Central). California and Texas (specified members) are excluded because they do not have descendants.

```
FIX
(@LDESCENDANTS(@ATTRIBUTE(Caffeinated_True), @ATTRIBUTE(Ounces_12), "200-40"))
...
ENDFIX;
```

Returns an empty list as none of the specified members (100-10, 100-20, 200-10, 300-30, which are caffeinated, 12-ounce drinks, and 200-40) have descendants.

See Also

- [@ILDESCENDANTS](#)
- [@IDDESCENDANTS](#)
- [@DESCENDANTS](#)

@LEV

Returns the level number of the specified member.

Syntax

@LEV (*mbrName*)

Parameter Description

mbrName Any valid single member name or member combination, or a function that returns a single member or member combination.

Example

In the Sample Basic database:

Function	Level Returned
@LEV(Margin)	1
@LEV("New York")	0

See Also

- [@CURLEV](#)
- [@GEN](#)

@LEVMBRS

Returns all members with the specified level number or level name in the specified dimension.

Syntax

@LEVMBRS (*dimName*, *levName/levNum*)

Parameter	Description
dimName	Dimension name specification.
levName levNum	A level name or an integer value that defines the number of a level. The integer value must be 0 or a positive integer.

Notes

- If you specify a name for the *levName* parameter, Essbase looks for a level with that name in the specified dimension.
- If you specify a number for the *levName* parameter (for example, 2), Essbase first looks for a level with a number string name. If no level name exists with that name, Essbase checks to see if the parameter is a valid level number.
- If you specify a temporary variable for the *levName* parameter, Essbase does not recognize the value of the variable. It looks in the outline for a level name with the same name as the temporary variable.
- For more information about levels and defining level names, see the *Oracle Essbase Database Administrator's Guide*.
- Essbase sorts the generated list of members in ascending order. Using Sample Basic as an example, if you specify `@LEVMBRS (Product, 1)`, Essbase returns 100, 200, 300, 400, Diet (in that order). This order is important to consider when you use the `@LEVMBRS` member set function with certain forecasting and statistical functions.
- If you use a negative number for the level number, no syntax error is noted, but the calculation will fail with an error message.

Example

In the Sample Basic database:

```
@LEVMBRS(Measures, "Profit and Loss")
@LEVMBRS (Measures, 0)
```

both return the following members if level 0 of the Measures dimension is named Profit and Loss:

Sales, COGS, Marketing, Payroll, Misc, Opening Inventory, Additions, Ending Inventory, Margin %, Profit %, and Profit per Ounce (in that order).

```
@LEVMBRS (Scenario, 0)
```

Returns Actual, Budget, Variance, and Variance %.

The following example restricts the calculation to members with the combination Budget and one of the members of the Market dimension with a level name of "State".

```
FIX (Budget, @LEVMBRS (Market, State))
  CALC DIM (Year, Measures);
ENDFIX
```

See Also

- [@GENMBRS](#)

@LIKE

Returns a member set of member names that match the specified pattern.

This function can be used on unique and duplicate-name outlines.

Syntax

```
@LIKE(pattern, topMbrinHierarchy, [escChar])
```

Parameter	Description
pattern	The character pattern with which to compare to members in the outline, including a single wildcard character: <ul style="list-style-type: none">• %: The percentage sign allows matching to a string of any length (including zero length).• _: The underscore allows matching on a single character in a member name.
topMbrinHierarchy	A fully qualified member name on which to base the member search. The specified member and its aliases, and all of its descendants, are included in the search. To search the entire outline, provide an empty string (" ") for this parameter. For example, @LIKE ("100%", " ").
escChar	Optional: A one-byte-length escape character to use if the wildcard character exists in member names. If you do not specify an escape character, a backslash (\) is assumed.

Example

The following examples are based on the following duplicate-name outline:

```
Product
  100
    100-10
      100-10-10
    100-20
    100-30
  200
    200-10
    200-20
    200-30
  300
    300-10
    300-20
Diet
  100-10
    100-10-11
  200-10
  300-10
Bottle
  200-10
  300-20
```

```
@LIKE("100%", "Product")
```

Returns members 100, 100-10, 100-20, and 100-30.

```
@LIKE("30_", "Product")
```

Returns member 300.

```
@LIKE("200\_", "Product", "\")
```

If member 200 has children named 200_10 (note the underscore, _), 200-20 (note the dash, -), 200_30 and 200-40, returns those members whose name contains an underscore: 200_10 and 200_30.

See Also

- [@BETWEEN](#)
- [@EQUAL](#)
- [@EXPAND](#)
- [@MBRCOMPARE](#)
- [@MBRPARENT](#)
- [@NOTEQUAL](#)

@LIST

Creates and distinguishes lists that are processed by functions that require list arguments. @LIST can be used to create *expLists*, member lists, or *rangeLists*. @LIST treats a collection of parameters as one entity.

Syntax

```
@LIST (argument1, argument2, ..., argumentN)
```

Parameter	Description
argument1, argument2, ..., argumentN	The list of arguments that are collected and treated as one argument so they can be processed by the parent function. Arguments can be member names, member combinations, member set functions, range functions, and numeric expressions.

Notes

@LIST does not check for or eliminate duplicates.

Example

The following example is based on the Sample Basic database. Assume that the Year dimension contains an additional member, Sales Correl. @LIST is used with the @CORRELATION function to determine the sales relationship between a product's two peak periods (Jan through Mar and Apr through May):

```
FIX(Sales)
"Sales Correl" = @CORRELATION(SKIPNONE,
    @LIST(Jan, Feb, Mar) , @LIST(Apr, May, Jun) ) ;
ENDFIX
```

This example produces the following report:

```
          Colas      Actual      New York
                Sales
                =====
Jan            678
```

Feb	645
Mar	675
Apr	712
May	756
Jun	890

Sales Correl 0.200368468

@LN

Returns the natural logarithm (base e) of the specified expression.

Syntax

@LN (*expression*)

Parameter Description

expression Single member specification, member combination, or other numeric expression. If less than or equal to 0, Essbase returns #MISSING.

Example

The following example is based on a variation of Sample Basic:

```
LN_Sales = @LN(Sales);
```

This example produces the following result:

	Cola East				
	Jan	Feb	Mar	Nov	Dec
Sales	100	110	120 . . .	0	210
LN_Sales	4.65052	4.70048	4.78749 . . .	#MISSING	5.34710

See Also

- [@LOG10](#)
- [@LOG](#)
- [@EXP](#)

@LOG

Returns the result of a logarithm calculation where you can specify both the base to use and the expression to calculate.

Syntax

@LOG (*expression* [, *base*])

Parameter Description

expression Single member specification, variable name, function, or other numeric expression. If less than or equal to 0, Essbase returns #MISSING.

Parameter Description

- base Optional. Single member specification, member combination, or numeric expression.
- If the base value is #MISSING, less than or equal to 0, or close to 1, Essbase returns #MISSING.
 - If the base is omitted, Essbase calculates the base-10 logarithm of the specified expression. @LOG(Sales) is equivalent to @LOG10(Sales).

Notes

The @LOG function returns the logarithm of *expression* calculated using the specified *base*. @LOG(x,b) is equivalent to $\log_b(x)$.

Example

The following example is based on a variation of Sample Basic:

```
LOG2_Sales = @LOG(Sales,2);
```

This example produces the following result:

	Cola East				
	Jan	Feb	Mar	Nov	Dec
Sales	100	#MISSING	120 . . .	0	210
LOG2_Sales	6.64386	#MISSING	6.90689 . . .	#MISSING	7.71425

See Also

- [@LN](#)
- [@LOG10](#)

@LOG10

Returns the base-10 logarithm of the specified expression.

Syntax

```
@LOG10 (expression)
```

Parameter Description

- expression Single member specification, variable name, function, or other numeric expression. If less than or equal to 0, Essbase returns #MISSING.

Example

The following example is based on a variation of Sample Basic:

```
LOG10_Sales = @LOG10(Sales);
```

This example produces the following result:

	Product			
	East	West	South	Central
Sales	87398	132931	50846	129680
LOG10_Sales	4.94150	5.12363	4.70626	5.11287

See Also

- [@LOG](#)
- [@LN](#)

@LSIBLINGS

Returns the left siblings of the specified member.

Syntax

@LSIBLINGS (*mbrName*)

Parameter Description

mbrName Any valid single member name or member combination, or a function that returns a single member or member combination.

Notes

This function returns the left siblings of the specified member. Left siblings are children that share the same parent as the member and that precede the member in the database outline. This function excludes the specified member.

This member set function can be used as a parameter of another function, where that parameter is a list of members.

Essbase sorts the generated list of left siblings in descending order. Using Sample Basic as an example, if you specify 200-30 for *mbrName*, Essbase returns 200-20, 200-10 (in that order). This order is important to consider when you use the @LSIBLINGS member set function with certain forecasting and statistical functions.

Example

In the Sample Basic database:

```
@LSIBLINGS(Qtr4)
```

Returns Qtr3, Qtr2, and Qtr1 (in that order). These members appear above Qtr4 in the Sample Basic outline.

```
@LSIBLINGS(Utah)
```

Returns Washington, Oregon, and California (in that order). These members appear above Utah in the Sample Basic outline.

See Also

- [@ILSIBLINGS](#)
- [@RSIBLINGS](#)
- [@NEXTSIBLING](#)
- [@PREVSIBLING](#)
- [@SHIFTSIBLING](#)

@MATCH

Performs wildcard member selections.

Syntax

```
@MATCH (mbrName|genName|levName, "pattern")
```

Parameter Description

mbrName	The default or user-defined name of the member on which to base the search. Essbase searches the member names and alias names of the specified member and its descendants.
genName	The default or user-defined name of the generation to search. Essbase searches all member names and member alias names in the generation.
levName	The default or user-defined name of the level to search. Essbase searches all member names and member alias names in the level.
"pattern"	The character pattern to search for, including a wildcard character (* or ?). ? substitutes one occurrence of any character. You can use ? anywhere in the pattern. * substitutes any number of characters. You can use * only at the end of the pattern. To include spaces in the character pattern, enclose the pattern in double quotation marks ("").

Notes

This function performs a trailing-wildcard member selection. Essbase searches for member names and alias names that match the pattern you specify and returns the member and alias names it finds.

If the members names in the database you are searching are case-sensitive, the search is case-sensitive. Otherwise, the search is not case-sensitive. To define database member names as case-sensitive, use Outline Editor in Oracle Essbase Administration Services. See the *Oracle Essbase Administration Services Online Help*.

You can use more than one @MATCH function in a calculation script.

If Essbase does not find any members that match the chosen character pattern, it returns no member names and continues with the other calculation commands in the calculation script.

Example

In the Sample Basic database:

```
@MATCH(Product, "???-10")
```

Returns 100-10, 200-10, 300-10, and 400-10

```
@MATCH(Year, "J*")
```

Returns Jan, Jun, Jul

```
@MATCH(Product, "C*")
```

Returns 100 (Colas), 100-10 (Cola), 100-30 (Caffeine Free Cola), 300 (Cream Soda)

@MAX

Returns the maximum value among the results of the expressions in the specified member list.

Syntax

```
@MAX (expList)
```

Parameter Description

expList Comma-delimited list of members, variable names, functions, and numeric expressions, all of which return numeric values.

Notes

Depending on the values in the list, @MAX may return a zero(0) or #MISSING value. For full control over skipping or inclusion of zero(0) and #MISSING values, it is recommended to use the @MAXS function instead of the @MAX function.

Example

This example is based on the Sample Basic database:

```
Qtr1 = @MAX(Jan:Mar);
```

This example produces the following report:

	Colas	New York	Actual	
	Jan	Feb	Mar	Qtr1
	===	===	===	====
Sales	678	645	675	678

See Also

- [@MAXS](#)
- [@MAXSRANGE](#)
- [@MINS](#)

@MAXRANGE

Returns the maximum value of the specified member across the specified range of members.

Syntax

```
@MAXRANGE (mbrName [ , XrangeList])
```

Parameter Description

mbrName Any valid single member name or member combination, or a function that returns a single member or member combination.

XrangeList A valid member name, a comma-delimited list of member names, cross-dimension members, or a member set function or range function (including @XRANGE) that returns a list of members from the same dimension. If *XrangeList* is not specified, Essbase uses the level 0 members from the dimension tagged as time.

Notes

Depending on the values in the list, @MAXRANGE may return a zero(0) or #MISSING value. For full control over skipping or inclusion of zero(0) and #MISSING values, it is recommended to use @MAXSRANGE instead of @MAXRANGE.

Example

In the Sample Basic database:

```
Qtr1 = @MAXRANGE(Sales, @CHILDREN(Qtr1));
```

produces the following report:

	Colas	New York	Actual	
	Jan	Feb	Mar	Qtr1
	===	===	===	====
Sales	678	645	675	678

See Also

- [@MAXSRANGE](#)
- [@MAXS](#)
- [@MINSRANGE](#)

@MAXS

Returns the maximum value among the results of the expressions in the specified member list, with options to skip missing or zero values (in contrast with the @MAX function, which cannot ignore these values).

Syntax

```
@MAXS (SKIPNONE | SKIPMISSING | SKIPZERO | SKIPBOTH, expList)
```

Parameter	Description
-----------	-------------

SKIPNONE	Includes all cells specified in <i>expList</i> in the operation, regardless of their content
----------	----------------------------------------------------------------------------------------------

SKIPMISSING	Ignores all #MISSING values
-------------	-----------------------------

SKIPZERO	Ignores all 0 values
----------	----------------------

SKIPBOTH	Ignores all 0 and #MISSING values
----------	-----------------------------------

<i>expList</i>	Comma-delimited list of members, variable names, functions, or numeric expressions, all of which return numeric values
----------------	------------------------------------------------------------------------------------------------------------------------

Notes

- @MAXS (SKIPMISSING, *expList*) is equivalent to @MAX (*expList*).
- Because #MISSING values are greater than negative data values and less than positive data values, if the data being calculated includes only negative and #MISSING values, @MAXS returns #MISSING.

- If the data being calculated includes only negative, 0, and #MISSING values, @MAXS may return either #MISSING or 0 values in an unpredictable manner.

Example

For both examples, assume a database similar to Sample Basic. The Measures dimension includes two members: COGS (cost of goods sold) and OtherInc_Exp (miscellaneous income and expenses). The data can include 0 and #MISSING values.

Example 1

```
Qtr1_Max = @MAXS(SKIPBOTH, Jan:Mar);
```

This example ignores #MISSING and 0 values for all members of the Measures dimension. This example produces the following results:

	Jan	Feb	Mar	Qtr1_Max
	=====	=====	=====	=====
COGS	#MISSING	1500	2300	2300
OtherInc_Exp	-500	-350	0	-350

Example 2

```
Qtr1_Max = @MAXS(SKIPNONE, Jan:Mar);
```

This example includes #MISSING and 0 values in the calculation, for all members of the Measures dimension. This example produces the following results:

	Jan	Feb	Mar	Qtr1_Max
	=====	=====	=====	=====
COGS	#MISSING	1500	2300	2300
OtherInc_Exp	-500	-350	0	0

See Also

- [@MAXSRANGE](#)
- [@MAX](#)
- [@MINS](#)

@MAXSRANGE

Returns the maximum value of the specified member across the specified range of members, with options to skip missing or zero values (in contrast with the @MAXRANGE function, which cannot ignore these values).

Syntax

```
@MAXSRANGE (SKIPNONE | SKIPMISSING | SKIPZERO | SKIPBOTH, mbrName [ ,XrangeList])
```

Parameter	Description
SKIPNONE	Includes all cells specified in <i>expList</i> in the operation, regardless of their content
SKIPMISSING	Ignores all #MISSING values

Parameter	Description
SKIPZERO	Ignores all 0 values
SKIPBOTH	Ignores all 0 and #MISSING values
mbrName	Any valid single member name or member combination, or a function that returns a single member or member combination
XrangeList	A valid member name, a comma-delimited list of member names, cross-dimension members, or a member set function or range function (including @XRANGE) that returns a list of members from the same dimension. If <i>XrangeList</i> is not specified, Essbase uses the level 0 members from the dimension tagged as time.

Notes

- @MAXSRANGE (SKIPNONE, *mbrName*, *XrangeList*) is equivalent to @MAXRANGE *mbrName*, (*XrangeList*).
- #MISSING values are considered to be greater than negative data values and less than positive data values. If the data being calculated includes only negative and #MISSING values, @MAXSRANGE returns #MISSING.
- For all members, @MAXSRANGE returns the value calculated for the specified member and range list.

Example

For both examples, assume a database similar to Sample Basic. The Measures dimension includes two members: COGS (cost of goods sold) and OtherInc_Exp (miscellaneous income and expenses). The data can include 0 and #MISSING values. For both members of the Measures dimension, the result is the same--the maximum value for the OtherInc_Exp member across the specified range.

Example 1

```
Qtr1_Max = @MAXSRANGE (SKIPBOTH, OtherInc_Exp, @CHILDREN(Qtr1));
```

This example ignores #MISSING and 0 values and produces the following results:

	Jan	Feb	Mar	Qtr1_Max
OtherInc_Exp	-500	#MISSING	-250	-250
COGS	0	1500	2300	-250

Example 2

```
Qtr1_Max = @MAXSRANGE (SKIPNONE, OtherInc_Exp, @CHILDREN(Qtr1));
```

Using the same data as Example 1, Example 2 demonstrates what happens if you do not skip 0 and #MISSING values in the data. Example 2 produces the following report:

	Jan	Feb	Mar	Qtr1_Max
OtherInc_Exp	-500	#MISSING	-250	#MISSING
COGS	0	1500	2300	#MISSING

See Also

- [@MAXS](#)
- [@MINSRANGE](#)
- [@MAXRANGE](#)

@MBRCOMPARE

Returns a member set of member names that match the comparison criteria. Member names are evaluated alphanumerically.

This function can be used on unique and duplicate-name outlines.

Syntax

```
@MBRCOMPARE (compOperator, tokenString, topMbrinHierarchy, cdfName)
```

Parameter	Description
compOperator	One of the following strings: < (less than), <= (less than or equal to), > (greater than), >= (greater than or equal to), == (equals), != (not equal to), or CDF (for a custom-defined function). Note: Using the == (equal to) comparison operator is the same as using the @EQUAL function. Using the != (not equal to) comparison operator is the same as using the @NOTEQUAL function.
tokenString	Token string value with which to compare to members in the outline, starting with the member specified in <i>topMbrinHierarchy</i> .
topMbrinHierarchy	A fully qualified name of a member in the outline on which to base the member search. The specified member and its aliases, and all of its descendants, are included in the search. Note: Although aliases of the specified member are included in the search, only outline member names (not aliases) are used when comparing member names. To search the entire outline, provide an empty string (" ") for this parameter. For example, @MBRCOMPARE ("<=" , "100-10" , " ").
cdfName	Optional: This argument applies only if CDF is specified for <i>compOperator</i> . Name of a custom-defined function. The custom-defined function must take the <i>tokenString</i> and <i>topMbrinHierarchy</i> arguments and return a Boolean value. (When compiling @MBRCOMPARE, Essbase rejects custom-defined functions that do not meet these requirements.) If the function returns a value of TRUE, the member is added to the member set returned by @MBRCOMPARE.

Notes

The following example of a custom-defined function returns results similar to using the >= (greater than or equal to) comparison operator:

```
package com.hyperion.essbase.cdf.comparecdf;  
  
class MyCDF {
```

```

public static boolean JavaNameCompare(String baseStr,
                                      String newStr)
{
try {
    System.out.println ("\n COMPARING MEMBER NAMES ..... \n ");
    // Compare the two strings.
    int result = newStr.compareToIgnoreCase(baseStr);
    if (result < 0)
        return false;
    else if (result == 0)
        return true;
    else
        return true;
}
catch (Exception e) {
    System.out.println ("Comparison function failed !!. Exception \n ");
    return false;
}
}

```

You must register the custom-defined function before you can use it in the @MBRCOMPARE function.

► To register the custom-defined function:

1 Compile the custom-defined function into a JAR file. For example:

```
CompareCDF.jar
```

2 Copy the JAR file to the following directory:

```
$ARBORPATH/java/udf
```

3 To grant access to the JAR file, add the following statement to the end of the udf.policy file, which is located in the \$ARBORPATH/java directory:

```
grant codeBase "file:${essbase.java.home}/../java/udf/ CompareCDF.jar" {
permission java.security.AllPermission;
};
```

4 To register the custom-defined function, use the following MaxL statement:

```
CREATE OR REPLACE FUNCTION '@JAVACOMPARE'
AS com.hyperion.essbase.cdf.comparecdf.MyCDF.JavaNameCompare(String,
String) '
SPEC '@ CUSTOMCOMPARE (Str1, Str2)'
COMMENT 'Compares Strings returns boolean flag';
```

Example

The following examples are based on the following duplicate-name outline:

```

Product
  100
    100-10
      100-10-10
    100-20
    100-30
  200
    200-10

```

```

    200-20
    200-30
300
    300-10
    300-20
Diet
    100-10
    100-10-11
    200-10
    300-10
Bottle
    200-10
    300-20

```

```
@MBRCOMPARE("<=", "100-10", "Product")
```

Returns the members 100, [100].[100-10], and [Diet].[100-10].

```
@MBRCOMPARE("==", "100-10", "Product")
```

Returns the members [Diet].[100-10] and [100].[100-10].

```
@MBRCOMPARE ("CDF", "100-20", "100", @JAVACOMPARE)
```

Uses the @JAVACOMPARE custom-defined function to return a member set.

See Also

- [@BETWEEN](#)
- [@EQUAL](#)
- [@EXPAND](#)
- [@LIKE](#)
- [@MBRPARENT](#)
- [@NOTEQUAL](#)

@MBRPARENT

Returns the parent of the specified member.

This function can be used on unique and duplicate-name outlines.

Syntax

```
@MBRPARENT (mbrName)
```

Parameter Description

mbrName Name of a member in the outline.

Example

The following examples are based on the following duplicate-name outline:

```

Product
  100
    100-10
      100-10-10

```

```

100-20
100-30
200
200-10
200-20
200-30
300
300-10
300-20
Diet
100-10
100-10-11
200-10
300-10
Bottle
200-10
300-20

```

```
@MBRPARENT ("100-10", "Product")
```

Returns the member 200.

```
@MBRPARENT ("100-10-11")
```

Returns the member [Diet].[100-10].

See Also

- [@BETWEEN](#)
- [@EQUAL](#)
- [@EXPAND](#)
- [@LIKE](#)
- [@MBRCOMPARE](#)
- [@NOTEQUAL](#)

@MDALLOCATE

Allocates values from a member, from a cross-dimensional member, or from a value across multiple dimensions. The allocation is based on a variety of criteria.

This function allocates values that are input at an upper level to lower-level members in multiple dimensions. The allocation is based upon a specified share or spread of another variable. You can specify a rounding parameter for allocated values and account for rounding errors.

Syntax

```
@MDALLOCATE (amount, Ndim, allocationRange1 ... allocationRangeN, basisMbr, [roundMbr],
method [, methodParams]
```

```
[, round [, numDigits][, roundErr]])
```


Parameter	Description
amount	<p>A value, member, or cross-dimensional member that contains the value to be allocated into each <i>allocationRange</i>. The value may also be a constant.</p> <ul style="list-style-type: none"> ● If <i>amount</i> is a member, the member must be from a dimension to which an <i>allocationRange</i> belongs. ● If <i>amount</i> is a cross-dimensional member, the member must include a member from every dimension of every <i>allocationRange</i>. ● If a member or cross-dimensional member is not from an <i>allocationRange</i> dimension, Essbase displays a warning message. <p>If the <i>amount</i> parameter is a loaded value, it cannot be a Dynamic Calc member.</p>
Ndim	The number of dimensions across which values are allocated.
allocationRange1 ... allocationRangeN	Comma-delimited lists of members, member set functions, or range functions from the multiple dimensions into which values from <i>amount</i> are allocated.
basisMbr	A value, member, or cross-dimensional member that contains the values that are used as the basis for the allocation. The <i>method</i> you specify determines how the basis data is used.
roundMbr	Optional. The member or cross-dimensional member to which rounding errors are added. This member (or at least one member of a cross-dimensional member) must be included in an <i>allocationRange</i> .

Parameter	Description
method	<p>The expression that determines how values are allocated. One of the following:</p> <ul style="list-style-type: none"> ● <i>share</i>: Uses <i>basisMbr</i> to calculate a percentage share. The percentage share is calculated by dividing the value in <i>basisMbr</i> for the current member in <i>allocationRange</i> by the sum across the <i>allocationRange</i> for that basis member: $\text{amount} * (\text{@CURRMBR}() \rightarrow \text{basisMbr} / \text{@SUM}(\text{allocationRange} \rightarrow \text{basisMbr}))$ ● <i>spread</i>: Spreads <i>amount</i> across <i>allocationRange</i>: $\text{amount} * (1 / \text{@COUNT}(\text{SKIP}, \text{allocationRange}))$ ● SKIPNONE SKIPMISSING SKIPZERO SKIPBOTH: Values to be ignored during calculation of the spread. You must specify a SKIP parameter only for <i>spread</i>. <ul style="list-style-type: none"> ○ SKIPNONE: Includes all cells. ○ SKIPMISSING: Excludes all #MISSING values in <i>basisMbr</i>, and stores #MISSING for values in <i>allocationRange</i> for which the <i>basisMbr</i> is missing. ○ SKIPZERO: Excludes all zero (0) values in <i>basisMbr</i>, and stores #MISSING for values in <i>allocationRange</i> for which the <i>basisMbr</i> is zero. ○ SKIPBOTH: Excludes all zero (0) values and all #MISSING values, and stores #MISSING for values in <i>allocationRange</i> for which the <i>basisMbr</i> is zero (0) or #MISSING. ● <i>percent</i>: Takes a percentage value from <i>basisMbr</i> for each member in <i>allocationRange</i> and applies the percentage value to <i>amount</i>: $\text{amount} * (\text{@CURRMBR}() \rightarrow \text{basisMbr} * .01).$ ● <i>add</i>: Takes the value from <i>basisMbr</i> for each member of <i>allocationRange</i> and adds the value to <i>amount</i>: $\text{amount} + \text{@CURRMBR}() \rightarrow \text{basisMbr}$ ● <i>subtract</i>: Takes the value from <i>basisMbr</i> for each member of <i>allocationRange</i> and subtracts the value from <i>amount</i>: $\text{amount} - \text{@CURRMBR}() \rightarrow \text{basisMbr}$ ● <i>multiply</i>: Takes the value from <i>basisMbr</i> for each member of <i>allocationRange</i> and multiplies the value by <i>amount</i>: $\text{amount} * \text{@CURRMBR}() \rightarrow \text{basisMbr}$ ● <i>divide</i>: Takes the value from <i>basisMbr</i> for each member of <i>allocationRange</i> and divides the value by <i>amount</i>: $\text{amount} / \text{@CURRMBR}() \rightarrow \text{basisMbr}$
round	<p>Optional. One of the following:</p> <ul style="list-style-type: none"> ● <i>noRound</i>: No rounding. <i>noRound</i> is the default. ● <i>roundAmt</i>: Indicates that you want to round the allocated values. If you specify <i>roundAmt</i>, you also must specify <i>numDigits</i> to indicate the number of decimal places to round to.

Parameter	Description
numDigits	<p>An integer that represents the number of decimal places to round to. You must specify <i>numDigits</i> if you specify <i>roundAmt</i>.</p> <ul style="list-style-type: none"> ● If <i>numDigits</i> is 0, the allocated values are rounded to the nearest integer. The default value for <i>numDigits</i> is 0. ● If <i>numDigits</i> is greater than 0, the allocated values are rounded to the specified number of decimal places. ● If <i>numDigits</i> is a negative value, the allocated values are rounded to a power of 10. <p>If you specify <i>roundAmt</i>, you also can specify a <i>roundErr</i> parameter.</p>
roundErr	<p>Optional. An expression that specifies where rounding errors should be placed. You must specify <i>roundAmt</i> in order to specify <i>roundErr</i>. If you do not specify <i>roundErr</i>, Essbase discards rounding errors.</p> <p>To specify <i>roundErr</i>, choose from one of the following:</p> <ul style="list-style-type: none"> ● <i>errorsToHigh</i>: Adds rounding errors to the member with the highest allocated value. If allocated values are identical, adds rounding errors to the first value in <i>allocationRange</i>. ● <i>errorsToLow</i>: Adds rounding errors to the member with the lowest allocated value. If allocated values are identical, adds rounding errors to the first value in <i>allocationRange</i>. #MISSING is treated as the lowest value in a list; if multiple values are #MISSING, rounding errors are added to the first #MISSING value in the list. ● <i>errorsToMbr</i>: Adds rounding errors to the specified <i>roundMbr</i>, which must be included in <i>allocationRange</i>.

Notes

- When you use @MDALLOCATE in a calculation script, use it within a FIX statement; for example, FIX on the member to which the allocation amount is loaded. Although FIX is not required, using it may decrease calculation time.
- For a more complex example using the @MDALLOCATE function, see the *Oracle Essbase Database Administrator's Guide*.
- If you have very large *allocationRange* lists, Essbase may return error messages during the calculation. If you receive error messages, you may need to raise the number for CALCLOCKBLOCK DEFAULT or use CALCLOCKBLOCK HIGH in your calculation script.

Example

Consider the following example from the Sample Basic database. A data value of 500 is loaded to Budget->Total Expenses->East for Jan and Colas. (For this example, assume that Total Expenses is not a Dynamic Calc member.)

You need to allocate the amount across each expense category for each child of East. The allocation for each child of East is based on the child's share of Total Expenses->Actual:

```
FIX("Total Expenses")
Budget = @MDALLOCATE(Budget->"Total Expenses"->East, 2,
    @CHILDREN(East), @CHILDREN("Total Expenses"), Actual, , share);
ENDFIX
```

This example produces the following report:

		Marketing	Jan Payroll	Colas Misc	Total Expenses
		=====	=====	=====	=====
Actual	New York	94	51	0	145
	Massachusetts	23	31	1	55
	Florida	53	54	0	107
	Connecticut	40	31	0	71
	New Hampshire	27	53	2	82
	East	237	220	3	460
Budget	New York	102.174	55.435	0	#MI
	Massachusetts	25	33.696	1.087	#MI
	Florida	57.609	58.696	0	#MI
	Connecticut	43.478	33.696	0	#MI
	New Hampshire	29.348	57.609	2.173	#MI
	East	#MI	#MI	#MI	500

See Also

- [@ALLOCATE](#)

@MDANCESTVAL

Returns ancestor-level data from multiple dimensions based on the current member being calculated.

Syntax

@MDANCESTVAL (*dimCount*, *dimName1*, *genLevNum1*. . . *dimNameX*, *genLevNumX* [,*mbrName*])

Parameter	Description
<i>dimCount</i>	Integer value that defines the number of dimensions from which ancestor values are being returned.
<i>dimName1</i> , . . . <i>dimNameX</i>	Defines the dimension names from which the ancestor values are to be returned. You must specify a <i>genLevNum</i> for every <i>dimName</i> .
<i>genLevNum</i> , . . . <i>genLevNumX</i>	Integer value that defines the absolute generation or level number from which the ancestor values are to be returned. A positive integer defines a generation reference. A negative number or value of 0 defines a level reference. You must specify a <i>dimName</i> for every <i>genLevNum</i> .
<i>mbrName</i>	Optional. Any valid single member name or member combination, or a function that returns a single member or member combination, from which the ancestor values are to be returned.

Example

Marketing expenses are captured at the Product Family and Region level in a product planning application. The Marketing Expense data must be allocated down to each Product code and State level based on Sales contribution. Data is captured as follows:

		Sales	Marketing
		=====	=====
New York	100-10	300	N/A
	100-20	200	N/A
	100	500	N/A
Boston	100-10	100	N/A
	100-20	400	N/A
	100	500	N/A

East	100-10	400	N/A
	100-20	600	N/A
	100	1000	200

The Marketing Expense value of 200 at East and Product code 100 is allocated down to each Product code and State with the following formula:

Marketing = (Sales / @MDANCESTVAL(2, Market, 2, Product, 2, Sales)) * @MDANCESTVAL(2, Market, 2, Product, 2, Marketing);

which produces the following result:

		Sales	Marketing
		=====	=====
New York	100-10	300	60
	100-20	200	40
	100	500	100
Boston	100-10	100	20
	100-20	400	80
	100	500	100
East	100-10	400	80
	100-20	600	120
	100	1000	200

The Marketing expenses can then be reconsolidated across Products and Markets.

See Also

- [@ANCESTVAL](#)
- [@SANCESTVAL](#)
- [@MDPARENTVAL](#)

@MDPARENTVAL

Returns parent-level data from multiple dimensions based on the current member being calculated.

Syntax

@MDPARENTVAL (*numDim*, *dimName1*, . . . *dimNameX* [, *mbrName*])

Parameter	Description
numDim	Integer value that defines the number of dimensions from which parent values are being returned.
dimName1, . . . dimNameX	Defines the dimension names from which the parent values are to be returned.
mbrName	Any valid single member name or member combination, or a function that returns a single member or member combination, from which the parent values are to be returned.

Example

Marketing expenses are captured at the Product Family and Region level in a product planning application. The Marketing Expense data must be allocated down to each Product code and State level based on Sales contribution.

Data is captured as follows:

		Sales	Marketing
		=====	=====
New York	100-10	300	N/A
	100-20	200	N/A
	100	500	N/A
Boston	100-10	100	N/A
	100-20	400	N/A
	100	500	N/A
East	100-10	400	N/A
	100-20	600	N/A
	100	1000	200

The Marketing Expense value of 200 at East and Product code 100 is allocated down to each Product code and State with the following formula:

```
Marketing = (Sales / @MDPARENTVAL(2, Market, Product, Sales)) * @MDPARENTVAL(2, Market, Product, Marketing);
```

which produces the following result:

		Sales	Marketing
		=====	=====
New York	100-10	300	60
	100-20	200	40
	100	500	N/A
Boston	100-10	100	20
	100-20	400	80
	100	500	N/A
East	100-10	400	N/A
	100-20	600	N/A
	100	1000	N/A

The Marketing expenses can then be reconsolidated across Products and Markets.

See Also

- [@PARENTVAL](#)
- [@SPARENTVAL](#)
- [@MDANCESTVAL](#)

@MDSHIFT

Shifts a series of data values across multiple dimension ranges.

Syntax

```
@MDSHIFT (mbrName, shiftCnt1, dimName1, [range1|(range1)], . . . shiftCntX, dimNameX, [rangeX|(rangeX)])
```

Parameter	Description
mbrName	Any valid single member name or member combination, or a function that returns a single member or member combination, from which the values are to be shifted.
shiftCnt1...shiftCntX	Integer that defines the number of member positions to shift.

Parameter	Description
dimName1, ... dimNameX	Defines the dimension names in which the shift is to occur.
range1 (range1) ... rangeX (rangeX)	Optional. A valid member name, a comma-delimited list of member names, member set functions, and range functions. If <i>rangeList</i> is not specified, Essbase uses the level 0 members from the dimension specified with the <i>dimName</i> parameter. If the range list is comma delimited, then the list must be enclosed in parentheses.

Example

The Budget figures for Ending Inventory need to be calculated by taking Prior Year->Opening Inventory results as a starting point:

		Jan	Feb	Mar	
		===	===	===	
Prior Year	Opening Inventory	110	120	130	. .
Budget	Ending Inventory	N/A	N/A	N/A	. .

The following calculation script assumes that the Scenario dimension is as follows:

```
Scenario
    Prior Year
    Budget

FIX (Budget)
"Ending Inventory" = @MDSHIFT("Opening Inventory", 1, Year, , -1, Scenario,);
ENDFIX
```

In this example, *range1* is not specified, so Essbase defaults to the level 0 members of the Year dimension, which was specified as the *dimName1* parameter. Since *range2* is also not specified, Essbase defaults to the level 0 members of the Scenario dimension, which was specified as the *dimName2* parameter. This example produces the following result:

		Jan	Feb	Mar	
		===	===	===	
Prior Year	Opening Inventory	110	120	130	. .
Budget	Ending Inventory	120	130	140	. .

See Also

- [@SHIFT](#)

@MEDIAN

Returns the median (the middle number) of the specified data set (*expList*). Half the numbers in the data set are larger than the median, and half are smaller.

Syntax

```
@MEDIAN (SKIPNONE | SKIPMISSING | SKIPZERO | SKIPBOTH, expList)
```

Parameter	Description
SKIPNONE	Includes all cells specified in <i>expList</i> , regardless of their content, during calculation of the median.
SKIPMISSING	Excludes all #MISSING values from <i>expList</i> during calculation of the median.
SKIPZERO	Excludes all zero (0) values from <i>expList</i> during calculation of the median.
SKIPBOTH	Excludes all zero (0) values and #MISSING values from <i>expList</i> during calculation of the median.
<i>expList</i>	Comma-delimited list of member specifications, variable names, functions, or numeric expressions. <i>expList</i> provides a list of numeric values across which the median is calculated.

Notes

- If the member you are calculating and *expList* are not in the same dimension, use the @RANGE function to cross the member with the list of members (for example, to cross Sales with the children of 100).
- @MEDIAN sorts *expList* in ascending order before calculating the median.
- When *expList* contains an even number of values, the @MEDIAN function calculates the average of the two middle numbers.
- @MEDIAN treats #MISSING values as 0 unless SKIPMISSING or SKIPBOTH is specified.
- When you use @MEDIAN in a calculation script, use it within a FIX statement. Although FIX is not required, using it may improve calculation performance.
- When you use @MEDIAN across a large range in a sparse dimension, you may need to increase the size of the calculator cache. For more information on the calculator cache, see the *Oracle Essbase Database Administrator's Guide*.

Example

The following example is based on the Sample Basic database. Assume that the Measures dimension contains an additional member, Median. This example calculates the median sales values for all products and uses the @RANGE function to generate *expList*:

```
FIX (Product)
Median = @MEDIAN(SKIPBOTH, @RANGE(Sales, @CHILDREN(Product)));
ENDFIX
```

This example produces the following report:

		Jan	New York
		Actual	Budget
		=====	=====
Sales	Colas	678	640
	Root Beer	551	530
	Cream Soda	663	510
	Fruit Soda	587	620
	Diet Drinks	#MI	#MI
	Product	2479	2300
Median	Product	625	575

Because SKIPBOTH is specified in the calculation script, the #MI values for Diet Drinks are skipped. The remaining four products create an even-numbered data set. So, to calculate Median->Product->Actual, the two middle numbers in the set (587 and 663) are averaged to create the median (625). To calculate Median->Product->Budget, the two middle numbers in the set (530 and 620) are averaged to create the median (575).

See Also

- [@RANGE](#)

@MEMBER

Returns the member with the name that is provided as a character string.

Syntax

```
@MEMBER (String)
```

Parameter Description

String A string (enclosed in double quotation marks) or a function that returns a string

Example

Typically, the @MEMBER function is used in combination with string functions that are used to manipulate character strings to form the name of a member. In the following example, the member name QTR1 is appended to the character string 2000_ to form the string 2000_QTR1. The @MEMBER function returns the member 2000_QTR1 and QTD is set to the value of this member.

```
QTD=@MEMBER (@CONCATENATE ("2000_", QTR1));
```

See Also

- [@CONCATENATE](#)
- [@SUBSTRING](#)

@MERGE

Merges two member lists that are processed by another function. Duplicates (values found in both lists) are included only once in the merged list.

Syntax

```
@MERGE (list1, list2)
```

Parameter Description

list1 The first list of member specifications to be merged.

list2 The second list of member specifications to be merged.

Notes

- Duplicate values are included only once in the merged list.
- @MERGE can merge only two lists at a time. You can nest @MERGE functions to merge more than two lists.

Example

Example 1

In the Sample Basic database,

```
@MERGE(@CHILDREN(Colas),@CHILDREN("Diet Drinks"));
```

returns Cola, Diet Cola, Caffeine Free Cola, Diet Root Beer, and Diet Cream Soda.

Diet Cola appears only once in the merged list, even though it is a child of both Colas and Diet Drinks.

Example 2

In this example, the @MERGE function is used with the @ISMBR function to increase the marketing budget for major markets and for western markets.

```
Budget
(IF (@ISMBR(@MERGE(@UDA(Market,"Major Market"),
@DESCENDANTS(West))))
Marketing = Marketing * 1.1;
ENDIF);
```

This example produces the following report, which shows only the major markets in the East and all western markets:

Product	Year Marketing =====	Budget
New York	6039	
Massachusetts	1276	
Florida	2530	
California	7260	
Oregon	2090	
Washington	2772	
Utah	1837	
Nevada	4521	

The values prior to running the calculation script were:

New York	5490
Massachusetts	1160
Florida	2300
California	6600
Oregon	1900
Washington	2520
Utah	1670
Nevada	4110

See Also

- [@LIST](#)
- [@RANGE](#)
- [@REMOVE](#)

@MIN

Returns the minimum value among the results of the expressions in *expList*.

Syntax

```
@MIN (expList)
```

Parameter Description

expList Comma-delimited list of members, variable names, functions, and numeric expressions, all of which return numeric values.

Notes

Depending on the values in the list, @MIN may return a zero(0) or #MISSING value. For full control over skipping or inclusion of zero(0) and #MISSING values, it is recommended to use the [@MINS](#) function instead of the @MIN function.

Example

In the Sample Basic database:

```
Qtr1 = @MIN(Jan:Mar);
```

produces the following report:

	Colas	New York		Actual
	Jan	Feb	Mar	Qtr1
	===	===	===	====
Sales	678	645	675	645

See Also

- [@MINS](#)
- [@MINRANGE](#)
- [@MAX](#)

@MINRANGE

Returns the minimum value of *mbrName* across *XrangeList*.

Syntax

```
@MINRANGE (mbrName [ , XrangeList])
```

Parameter Description

mbrName	Any valid single member name or member combination, or a function that returns a single member or member combination.
XrangeList	Optional. A valid member name, a comma-delimited list of member names, cross dimension members, or a member set function or range function (including @XRANGE) that returns a list of members from the same dimension. If <i>XrangeList</i> is not specified, Essbase uses the level 0 members from the dimension tagged as time.

Notes

Depending on the values in the list, @MINRANGE may return a zero(0) or #MISSING value. For full control over skipping or inclusion of zero(0) and #MISSING values, it is recommended to use the @MINSRANGE function instead of the @MINRANGE function.

Example

In the Sample Basic database:

```
Qtr1 = @MINRANGE(Sales, Jan:Mar);
```

produces the following report:

	Colas	New York	Actual	
	Jan	Feb	Mar	Qtr1
	===	===	===	====
Sales	678	645	675	645

See Also

- [@MINSRANGE](#)
- [@MIN](#)
- [@MAXSRANGE](#)

@MINS

Returns the minimum value across the results of the expressions in *expList*, with options to skip missing or zero values.

Syntax

```
@MINS (SKIPNONE | SKIPMISSING | SKIPZERO | SKIPBOTH, expList)
```

Parameter Description

SKIPNONE	Includes in the operation all cells specified in <i>expList</i> regardless of their content
SKIPMISSING	Ignores all #MISSING values
SKIPZERO	Ignores all 0 values
SKIPBOTH	Ignores all 0 and #MISSING values
<i>expList</i>	Comma-delimited list of member names, variable names, functions, or numeric expressions. <i>expList</i> provides a list of numeric values for which Essbase determines the minimum value.

Notes

- @MINS enables skipping of #MISSING and 0 values, in contrast with @MIN, which always includes these values.
- @MINS (SKIPNONE, *expList*) is equivalent to @MIN (*expList*).
- Because #MISSING values are less than positive data values and more than negative data values, if the data being calculated includes only positive and #MISSING values, @MINS returns #MISSING.
- If the data being calculated includes only negative, 0, and #MISSING values, @MINS may return either #MISSING or 0 values in an unpredictable manner.

Example

For both examples, assume a database similar to Sample Basic. The Measures dimension includes two members: COGS (cost of goods sold) and OtherInc_Exp (miscellaneous income and expenses). The data can include 0 and #MISSING values.

Example 1

```
Qtr1_Min = @MINS(SKIPBOTH, Jan:Mar);
```

This example ignores #MISSING and 0 values for all members of the Measures dimension. This example produces the following results:

	Jan	Feb	Mar	Qtr1_Min
	=====	=====	=====	=====
COGS	#MISSING	1500	2300	1500
OtherInc_Exp	-500	-350	0	-500

Example 2

```
Qtr1_Min = @MINS(SKIPNONE, Jan:Mar);
```

For all members of the Measures dimension, this example includes #MISSING and 0 values and produces the following results:

	Jan	Feb	Mar	Qtr1_Min
	=====	=====	=====	=====
COGS	#MISSING	1500	2300	#MISSING
OtherInc_Exp	-500	-350	0	-500

See Also

- [@MINSRANGE](#)
- [@MAXS](#)
- [@MIN](#)

@MINSRANGE

Returns the minimum value of *mbrName* across *XrangeList*, with options to skip missing or zero values.

Syntax

@MINSRANGE (SKIPNONE | SKIPMISSING | SKIPZERO | SKIPBOTH, *mbrName* [, *XrangeList*])

Parameter	Description
SKIPNONE	Includes in the operation all specified cells regardless of their content
SKIPMISSING	Ignores all #MISSING values
SKIPZERO	Ignores all 0 values
SKIPBOTH	Ignores all 0 and #MISSING values
<i>mbrName</i>	Any valid single member name or member combination, or a function that returns a single member or member combination
<i>XrangeList</i>	Optional. A valid member name, a comma-delimited list of member names, cross dimension members, or a member set function or range function (including @XRANGE) that returns a list of members from the same dimension. If <i>XrangeList</i> is not specified, Essbase uses the level 0 members from the dimension tagged as time.

Notes

- @MINSRANGE enables skipping of #MISSING and 0 values, in contrast with the @MINRANGE function, which always includes these values in the calculation.
- @MINSRANGE (SKIPNONE, *mbrName*, *rangeList*) is equivalent to @MINRANGE (*mbrName*, *rangeList*).
- #MISSING values are considered to be less than positive data values and more than negative data values. If the data being calculated includes only positive and #MISSING values, @MINSRANGE returns #MISSING.
- For all members, @MINSRANGE returns the value calculated for the specified member and range list.

Example

For both examples, assume a database similar to Sample Basic. The Measures dimension includes two members: COGS (cost of goods sold) and OtherInc_Exp (miscellaneous income and expenses). The data can include 0 and #MISSING values. For both members of the Measures dimension, the result is the same--the minimum value for the OtherInc_Exp member across the specified range.

Example 1

```
Qtr1_Min = @MINSRANGE(SKIPBOTH, OtherInc_Exp, Jan:Mar);
```

This example ignores the 0 value for Mar and produces the following results:

	Jan	Feb	Mar	Qtr1_Min
	=====	=====	=====	=====
COGS	#MISSING	1500	2300	350
OtherInc_Exp	500	350	0	350

Example 2

```
Qtr1_Min = @MINS(SKIPNONE, OtherInc_Exp, Jan:Mar);
```

This example does not ignore the 0 value in the calculation. This example produces the following results:

	Jan	Feb	Mar	Qtr1_Min
	=====	=====	=====	=====
COGS	#MISSING	1500	2300	0
OtherInc_Exp	500	350	0	0

See Also

- [@MINS](#)
- [@MINRANGE](#)
- [@MAXSRANGE](#)

@MOD

Calculates the modulus of a division operation.

Syntax

```
@MOD (mbrName1, mbrName2)
```

Parameter	Description
-----------	-------------

mbrName1 and mbrName2 Members from the same dimension whose modulus is to be calculated.

Example

The following example is based on the Sample Basic database. Assume that the Measures dimension contains an additional member, Factor. The modulus between Profit % and Margin % is calculated with the following formula:

```
Factor = @MOD("Margin %", "Profit %");
```

This example produces the following report:

	Market	Product	Scenario
	Margin %	Profit %	Factor
	=====	=====	=====
Jan	55.10	25.44	4.22
Feb	55.39	26.03	3.34
Mar	55.27	25.87	3.53

@MODE

Returns the mode (the most frequently occurring value) in the specified data set (*expList*).

Syntax

```
@MODE (SKIPNONE | SKIPMISSING | SKIPZERO | SKIPBOTH, expList)
```

Parameter	Description
SKIPNONE	Includes all cells specified in <i>expList</i> , regardless of their content, during calculation of the mode.
SKIPMISSING	Excludes all #MISSING values from <i>expList</i> during calculation of the mode.
SKIPZERO	Excludes all zero (0) values from <i>expList</i> during calculation of the mode.
SKIPBOTH	Excludes all zero (0) values and #MISSING values from <i>expList</i> during calculation of the mode.
expList	Comma-delimited list of member specifications, variable names, functions, or numeric expressions. <i>expList</i> provides a list of numeric values across which the mode is calculated.

Notes

- When two or more values in *expList* occur at the same frequency, Essbase sorts the list of values in ascending order and chooses the lowest value that occurs with the most frequency as the mode. For example, if *expList* contains [2,1,2,2,2,3,3,3,3], Essbase sorts the list as [1,2,2,2,2,3,3,3,3] and chooses the value [2] as the mode.
- If *expList* contains no duplicate values, the @MODE function returns the smallest value in the list as the mode. For example, if *expList* contains [2,4,7,10,14], @MODE returns 2 as the mode.
- If #MISSING is the mode of *expList*, @MODE returns #MISSING unless SKIPMISSING or SKIPBOTH is specified. If you specify SKIPMISSING or SKIPBOTH and all values in *expList* are #MISSING, @MODE returns #MISSING. If you specify SKIPZERO or SKIPBOTH and all values in *expList* are 0, @MODE returns #MISSING.
- When you use @MODE in a calculation script, use it within a FIX statement. Although FIX is not required, using it may improve calculation performance.
- When you use @MODE across a large range in a sparse dimension, you may need to increase the size of the calculator cache. For more information on the calculator cache, see the *Oracle Essbase Database Administrator's Guide*.

Example

The following example calculates the mode of the units sold for the Central region and uses the @RANGE function to generate *expList*:

```
FIX (Central)
"Mode" = @MODE(SKIPMISSING,
  @RANGE(Sales,@CHILDREN(Central)));
ENDFIX
```

This example produces the following report:

Units Sold	Colas	Actual Units Sold	Jan
		=====	
	Illinois	3	
	Ohio	2	
	Wisconsin	3	
	Missouri	#MI	
	Iowa	0	
	Colorado	6	

	Central	14
Mode	Central	3

See Also

- [@RANGE](#)

@MOVAVG

Applies a moving n -term average (mean) to an input data set. Each term in the set is replaced by a trailing mean of n terms, and the first terms (the $n-1$ terms) are copies of the input data. @MOVAVG modifies a data set for smoothing purposes.

Syntax

```
@MOVAVG (mbrName [, n [, XrangeList]])
```

Parameter Description

mbrName	Any valid single member name or member combination, or a function that returns a single member or member combination.
n	Optional. A positive integer value that represents the number of values to average. The default is 3.
XrangeList	Optional. A valid member name, a comma-delimited list of member names, cross dimension members, or a member set function or range function (including @XRANGE that returns a list of members from the same dimension. If <i>XrangeList</i> is not specified, Essbase uses the level 0 members from the dimension tagged as time.

Notes

- The @MOVAVG function calculates a trailing, rather than a centered, average. For example:

Trailing Average			Centered Average		
1	2	3	1	2	3
		2		2	

- While calculating the moving average, the @MOVAVG function skips #MISSING values and decreases the denominator accordingly. For example, if one value out of three is #MISSING, Essbase adds the remaining two values and divides the sum by two.
- If you use a member set function to generate a member list for the *XrangeList* parameter (for example, @SIBLINGS), to ensure correct results, consider the order in which Essbase sorts the generated member list. For more information, see the *Oracle Essbase Technical Reference* topic for the member set function you are using.
- When you use @MOVAVG in a calculation script, use it within a FIX statement. Although FIX is not required, using it may improve calculation performance.
- For periods where the width is undefined, the value is the same as for the source member. For example, you can't compute the moving average over the last three months for Jan and Feb because it doesn't exist. When this happens, Essbase simply copies the value for Jan and Feb for the moving average.

- When you use @MOVAVG across a large range in a sparse dimension, you may need to increase the size of the calculator cache. For more information on the calculator cache, see the *Oracle Essbase Database Administrator's Guide*.

Example

The following example is based on the Sample Basic database. Assume that the Measures dimension contains an additional member, Mov Avg.

```
"Mov Avg" = @MOVAVG(Sales,3,Jan:Jun);
```

In this example, the @MOVAVG function smooths sales data for the first six months of the year (Jan through Jun). The results of @MOVAVG can be used with the @TREND function to forecast average sales data for a holiday season (for example, October - December).

This example produces the following report:

	Colas	New York	Actual
	Sales	Mov Avg	
	=====	=====	
Jan	678	678	
Feb	645	645	
Mar	675	666	
Apr	712	677.3	
May	756	714.3	
Jun	890	786	

In this example, Essbase averages three values at a time for the moving average. The first two values (Jan, Feb) for Mov Avg and the first two values for Sales are the same. The value for Mar represents the trailing average of Jan, Feb, and Mar. The value for Apr represents the trailing average of Feb, Mar, and Apr. The remaining values represent the trailing average for each group of three values.

See Also

- [@MOVMAX](#)
- [@MOV MED](#)
- [@MOV MIN](#)
- [@MOV SUM](#)
- [@MOV SUM X](#)
- [@TREND](#)

@MOVMAX

Applies a moving n -term maximum (highest number) to an input data set. Each term in the set is replaced by a trailing maximum of n terms, and the first terms (the $n-1$ terms) are copies of the input data. @MOVMAX modifies a data set for smoothing purposes.

Syntax

```
@MOVMAX (mbrName [, n [, XrangeList]])
```

Parameter Description

mbrName	Any valid single member name or member combination, or a function that returns a single member or member combination.
n	Optional. A positive integer value that represents the number of values that are used to calculate the moving maximum. The default is 3.
XrangeList	Optional. A valid member name, a comma-delimited list of member names, member set functions, and range functions. If <i>XrangeList</i> is not specified, Essbase uses the level 0 members from the dimension tagged as Time.

Notes

- The @MOVMAX function calculates a trailing, rather than a centered, maximum. For example:

Trailing Maximum			Centered Maximum		
1	2	3	1	2	3
		3			3

- While calculating the moving maximum, @MOVMAX skips #MISSING values. For example, if one value out of four is #MISSING, @MOVMAX calculates the maximum of the remaining three values.
- If you use an Essbase member set function to generate a member list for the *XrangeList* parameter (for example, @SIBLINGS), to ensure correct results, consider the order in which Essbase sorts the generated member list. For more information, see the *Oracle Essbase Technical Reference* topic for the member set function you are using.
- When you use @MOVMAX in a calculation script, use it within a FIX statement. Although FIX is not required, using it may improve calculation performance.
- When you use @MOVMAX across a large range in a sparse dimension, you may need to increase the size of the calculator cache. For more information on the calculator cache, see the *Oracle Essbase Database Administrator's Guide*.

Example

The following example is based on the Sample Basic database. Assume that the Measures dimension contains an additional member, Mov Max.

```
"Mov Max" = @MOVMAX(Sales,3,Jan:Jun);
```

In this example, the @MOVMAX function smooths sales data for the first six months of the year (Jan through Jun). The results of @MOVMAX can be used with the @TREND function to forecast maximum sales data for a holiday season (for example, October - December).

This example produces the following report:

	Root Beer Sales	New York Mov Max	Actual
	=====	=====	
Jan	551	551	
Feb	641	641	
Mar	586	641	
Apr	630	641	

May	612	630
Jun	747	747

In this example, Essbase uses three values at a time to calculate the moving maximum. The first two values (Jan, Feb) for Mov Max and the first two values for Sales are the same. The value for Mar represents the trailing maximum of Jan, Feb, and Mar. The value for Apr represents the trailing maximum of Feb, Mar, and Apr. The remaining values represent the trailing maximum for each group of three values.

See Also

- [@MOVAVG](#)
- [@MOV MED](#)
- [@MOV MIN](#)
- [@MOV SUM](#)
- [@MOV SUM X](#)
- [@TREND](#)

@MOV MED

Applies a moving n -term median (middle number) to an input data set. Each term in the list is replaced by a trailing median of n terms, and the first terms (the $n-1$ terms) are copies of the input data. @MOV MED modifies a data set for smoothing purposes.

Syntax

```
@MOV MED (mbrName [, n [, XrangeList]])
```

Parameter Description

mbrName	Any valid single member name or member combination, or a function that returns a single member or member combination.
n	Optional. A positive integer value that represents the number of values that are used to calculate the moving median. The default is 3.
XrangeList	Optional. A valid member name, a comma-delimited list of member names, cross dimension members, or a member set function or range function (including @XRANGE) that returns a list of members from the same dimension. If <i>XrangeList</i> is not specified, Essbase uses the level 0 members from the dimension tagged as time.

Notes

- While calculating the moving median, the @MOV MED function skips #MISSING values. For example, if one value out of four is #MISSING, @MOV MED calculates the median of the remaining three values.
- The @MOV MED function calculates a trailing, rather than a centered, median. For example:

Trailing Median	Centered Median
1 2 3	1 2 3
2	2

- If the group of values being used to calculate the median contains an even number of values, the @MOV MED function averages the two numbers in the middle.

- If you use an Essbase member set function to generate a member list for the *XrangeList* parameter (for example, @SIBLINGS), to ensure correct results, consider the order in which Essbase sorts the generated member list. For more information, see the *Oracle Essbase Technical Reference* topic for the member set function you are using.
- When you use @MOV MED in a calculation script, use it within a FIX statement. Although FIX is not required, using it may improve calculation performance.
- When you use @MOV MED across a large range in a sparse dimension, you may need to increase the size of the calculator cache. For more information on the calculator cache, see the *Oracle Essbase Database Administrator's Guide*.

Example

The following example is based on the Sample Basic database. Assume that the Measures dimension contains an additional member, Mov Med.

```
"Mov Med" = @MOV MED(Sales,3,Jan:Jun);
```

In this example, the @MOV MED function smooths sales data for the first six months of the year (Jan through Jun). The results of @MOV MED could be used with the @TREND function to forecast sales data for a holiday season (for example, October - December).

This example produces the following report:

	Colas	New York	Actual
	Sales	Mov Med	
	=====	=====	
Jan	678	678	
Feb	645	645	
Mar	675	675	
Apr	712	675	
May	756	712	
Jun	890	756	

In this example, Essbase uses three values at a time to calculate the moving median. The first two values (Jan, Feb) for Mov Med are the same as the first two values for Sales. The value for Mar represents the trailing median of Jan, Feb, and Mar. The value for Apr represents the trailing median of Feb, Mar, and Apr. The remaining values represent the trailing median of each group of three values.

See Also

- [@MOVAVG](#)
- [@MOVMAX](#)
- [@MOVMIN](#)
- [@MOVSUM](#)
- [@MOVSUMX](#)
- [@TREND](#)

@MOVMIN

Applies a moving n -term minimum (lowest number) to an input data set. Each term in the list is replaced by a trailing minimum of n terms, and the first terms (the $n-1$ terms) are copies of the input data. @MOVMIN modifies a data set for smoothing purposes.

Syntax

```
@MOVMIN (mbrName [, n [, XrangeList]])
```

Parameter Description

mbrName	Any valid single member name or member combination, or a function that returns a single member or member combination.
n	Optional. A positive integer value that represents the number of values that are used to calculate the moving minimum. The default is 3.
XrangeList	Optional. A valid member name, a comma-delimited list of member names, cross dimension members, or a member set function or range function (including @XRANGE) that returns a list of members from the same dimension. If <i>XrangeList</i> is not specified, Essbase uses the level 0 members from the dimension tagged as time.

Notes

- While calculating the moving minimum, the @MOVMIN function skips #MISSING values. For example, if one value out of four is #MISSING, @MOVMIN calculates the minimum of the remaining three values.
- The @MOVMIN function calculates a trailing, rather than a centered, minimum. For example:

Trailing Minimum			Centered Minimum		
1	2	3	1	2	3
		1			1

- If you use a member set function to generate a member list for the *XrangeList* parameter (for example, @SIBLINGS), to ensure correct results, consider the order in which Essbase sorts the generated member list. For more information, see the *Oracle Essbase Technical Reference* topic for the member set function you are using.
- When you use @MOVMIN in a calculation script, use it within a FIX statement. Although FIX is not required, using it may improve calculation performance.
- When you use @MOVMIN across a large range in a sparse dimension, you may need to increase the size of the calculator cache. For more information on the calculator cache, see the *Oracle Essbase Database Administrator's Guide*.

Example

The following example is based on the Sample Basic database. Assume that the Measures dimension contains an additional member, Mov Min.

```
"Mov Min" = @MOVMIN(Sales,3,Jan:Jun);
```

In this example, the @MOVMIN function smooths sales data for the first six months of the year (Jan through Jun). The results of @MOVMIN can be used with the @TREND function to forecast minimum sales data for the holiday season (for example, October - December).

This example produces the following report:

	Colas	New York	Actual
	Sales	Mov Min	
	=====	=====	
Jan	678	678	
Feb	645	645	
Mar	675	645	
Apr	712	645	
May	756	675	
Jun	890	712	

In this example, Essbase uses three values at a time to calculate the moving minimum. The first two values (Jan, Feb) for Mov Min and the first two values for Sales are the same. The value for Mar represents the trailing minimum of Jan, Feb, and Mar. The value for Apr represents the trailing minimum of Feb, Mar, and Apr. The remaining values represent the trailing minimum for each group of three values.

See Also

- [@MOVAVG](#)
- [@MOVMAX](#)
- [@MOV MED](#)
- [@MOV SUM](#)
- [@MOV SUM X](#)
- [@TREND](#)

@MOV SUM

Applies a moving sum to the specified number of values in an input data set. @MOV SUM modifies a data set for smoothing purposes.

Syntax

```
@MOV SUM (mbrName [, n [, XrangeList]])
```

Parameter Description

mbrName	Any valid single member name or member combination, or a function that returns a single member or member combination.
n	Optional. A positive integer value that represents the number of values to sum. The default is 3.
XrangeList	Optional. A valid member name, a comma-delimited list of member names, cross dimension members, or a member set function or range function (including @XRANGE) that returns a list of members from the same dimension. If XrangeList is not specified, Essbase uses the level 0 members from the dimension tagged as time.

Notes

- For example, if you specify 3 members of the Time dimension in the Sample Basic database, @MOVSUM at Mar is the sum of the values for Jan, Feb, and Mar; @MOVSUM at Apr is the sum of the values for Feb, Mar, and Apr. However, Jan and Feb have no @MOVSUM value, and are called trailing members. Trailing members are copies of the input values. If you wish to assign different values to trailing members, use @MOVSUMX instead.
- The @MOVSUM function calculates a trailing, rather than a centered, sum. This example illustrates the difference:

Trailing Sum			Centered Sum		
1	2	3	1	2	3
		6			6

- While calculating the moving sum, @MOVSUM skips #MISSING values. For example, if one value out of three is #MISSING, Essbase adds the remaining two values.
- If you use an Essbase member set function to generate a member list for the *XrangeList* parameter (for example, @SIBLINGS), to ensure correct results, consider the order in which Essbase sorts the generated member list. For more information, see the *Oracle Essbase Technical Reference* topic for the member set function that you are using.
- When you use @MOVSUM in a calculation script, use it within a FIX statement. Although FIX is not required, using it may improve calculation performance.
- When you use @MOVSUM across a large range in a sparse dimension, you may need to increase the size of the calculator cache. For more information on the calculator cache, see the *Oracle Essbase Database Administrator's Guide*.

Example

The following example is based on the Sample Basic database. Assume that the Measures dimension contains an additional member, Mov Sum.

```
"Mov Sum" = @MOVSUM(Sales,3,Jan:Jun);
```

In this example, @MOVSUM smooths sales data for the first six months of the year (Jan through Jun). The results of @MOVSUM can be used with the @TREND function to forecast average sales data for a holiday season (for example, October through December).

This example produces the following report:

	Colas	New York	Actual
	Sales	Mov Sum	
	=====	=====	
Jan	678	678	
Feb	645	645	
Mar	675	1998	
Apr	712	2032	
May	756	2143	
Jun	890	2358	

See Also

- [@MOVAVG](#)
- [@MOVMAX](#)
- [@MOV MED](#)

- [@MOVMIN](#)
- [@MOVSUMX](#)
- [@TREND](#)

@MOVSUMX

Applies a moving sum to the specified number of values in an input data set. @MOVSUMX modifies a data set for smoothing purposes.

Unlike @MOVSUM, @MOVSUMX allows you to specify the values assigned to trailing members. For example, if you specify three members of the Time dimension in the Sample Basic database, @MOVSUMX at Mar is the sum of the values for Jan, Feb, and Mar; @MOVSUMX at Apr is the sum of the values for Feb, Mar, and Apr. However, Jan and Feb have no @MOVSUMX value, and are called *trailing members*.

Syntax

```
@MOVSUMX (COPYFORWARD | TRAILMISSING | TRAILSUM, mbrName [,n[,Xrangelist]] )
```

Parameter	Description
COPYFORWARD	Copies the input value into the trailing members. This behavior is the same as the @MOVSUM function.
TRAILMISSING	Sets the value of the trailing members to #MISSING.
TRAILSUM	Sums the trailing values.
mbrName	Any valid single member name or member combination, or a function that returns a single member or member combination.
n	Optional. A positive integer value that represents the number of values that are used to calculate the moving maximum. The default is 3.
XrangeList	Optional. A valid member name, a comma-delimited list of member names, cross dimension members, or a member set function or range function (including @XRANGE) that returns a list of members from the same dimension. If <i>XrangeList</i> is not specified, Essbase uses the level 0 members from the dimension tagged as time.

Notes

- The @MOVSUMX function calculates a trailing, rather than a centered, sum. This example illustrates the difference:

Trailing Sum			Centered Sum		
1	2	3	1	2	3
		6			6

- While calculating the moving sum, @MOVSUMX skips #MISSING values. For example, if one value out of three is #MISSING, Essbase adds the remaining two values.
- If you use a member set function to generate a member list for the *XrangeList* parameter (for example, @SIBLINGS), to ensure correct results, consider the order in which Essbase sorts the generated member list. For more information, see the *Oracle Essbase Technical Reference* topic for the member set function that you are using.

- When you use @MOVSUMX in a calculation script, use it within a FIX statement. Although FIX is not required, using it may improve calculation performance.
- When you use @MOVSUMX across a large range in a sparse dimension, you may need to increase the size of the calculator cache. For more information on the calculator cache, see the *Oracle Essbase Database Administrator's Guide*.

Example

The following examples are based on the Sample Basic database. Assume that the Measures dimension contains an additional member, "Last 3 Months of Sales," and that the original Sales values are as shown.

```
Last 3 Months of Sales = @MOVSUMX (COPYFORWARD,Sales,3,Jan:Aug);
```

or:

```
Last 3 Months of Sales = @MOVSUMX (TRAILMISSING,Sales,3,Jan:Aug);
```

or:

```
Last 3 Months of Sales = @MOVSUMX (TRAILSUM,Sales,3,Jan:Aug);
```

These examples produce the following reports:

Sales

```
=====
Jan      100
Feb      150
Mar      200
Apr      250
May      300
Jun      350
Jul      400
Aug      450
```

```
Last 3 Months of Sales
COPYFORWARD
```

```
=====
          100
          150
          450
          600
          750
          900
         1050
         1200
```

```
Last 3 Months of Sales
TRAILMISSING
```

```
=====
      #MISSING
      #MISSING
       450
       600
       750
       900
```

1050
1200

Last 3 Months of Sales
TRAILSUM

```
=====
      100
      250
      450
      600
      750
      900
     1050
     1200
```

See Also

- [@MOVAVG](#)
- [@MOVMAX](#)
- [@MOV MED](#)
- [@MOV MIN](#)
- [@MOV SUM](#)
- [@TREND](#)

@NAME

Passes the enclosed string, or list of member or dimension names, as a list of strings to another function.

Syntax

@NAME (*mbrName*)

Parameter Description

mbrName A list of member names, dimension names, or strings.

Notes

Essbase does not support strings in functions. It treats strings as values or an array of values. The @NAME function processes strings.

Example

Example 1

The following example is based on the Sample Basic database. A user-defined function is used to retrieve the price from the table below. The user defined function (J_GetPrice) takes two string parameters, time and product name, to return the price for each product.

MonthName	ProductId	Price
Jan	100-10	1.90

MonthName	ProductId	Price
Feb	100-10	1.95
Mar	100-10	1.98
Jan	100-20	1.95
Feb	100-20	2.00
Mar	100-20	2.05

```
Price = @J_GetPrice(@NAME(@CURRMBR(Product)), @NAME(@CURRMBR(Year)));
```

The following report illustrates the above example:

	Price	Actual	Market
	Jan	Feb	Mar
	===	===	===
100-10	1.90	1.95	1.98
100-20	1.95	2.00	2.05

Example 2

The following example is based on the Sample Basic database:

```
"Profit Per Ounce" = Profit/@ATTRIBUTEVAL(@NAME(Ounces));
```

The @NAME function processes the string "Ounces" before passing it to the @ATTRIBUTEVAL function. This example produces the following report:

	Actual	Year	West
	Profit	Profit	Per Ounce
	=====	=====	=====
Cola	4593	382.75	

See Also

- [@CURRMBR](#)

@NEXT

Returns the *n*th cell value in the sequence *rangeList* from *mbrName*, retaining all other members identical to the current member. @NEXT cannot operate outside the given range.

Syntax

```
@NEXT (mbrName [, n, rangeList])
```

Parameter Description

- | | |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| mbrName | Any valid single member name or member combination, or a function that returns a single member or member combination. |
| n | Optional signed integer. If you do not specify <i>n</i> , then the default is set to 1, which provides the next member in the range. Using a negative value for <i>n</i> has the same effect as using the matching positive value in the @PRIOR function. |

Parameter Description

rangeList Optional. A valid member name, a comma-delimited list of member names, member set functions, and range functions. If *rangeList* is not specified, Essbase uses the level 0 members from the dimension tagged as Time.

Example

In this example, Next Cash for each month is derived by taking the Cash value for the following month. Since *n* is not specified, the default is 1, which provides the next member in the range. Since *rangeList* is not specified, the level 0 members from the dimension tagged as Time are used (Jan, Feb, Mar, ...).

```
"Next Cash" = @NEXT(Cash);
```

This example produces the following report:

	Jan	Feb	Mar	Apr	May	Jun
	===	===	===	===	===	===
Cash	100	90	120	110	150	100
Next Cash	90	120	110	150	100	#MI

See Also

- [@PRIOR](#)
- [@SHIFT](#)
- [@SHIFTMINUS](#)
- [@SHIFTPLUS](#)

@NEXTS

Returns the *n*th cell value in the sequence *rangeList* from the *mbrName*. Provides the option to skip #MISSING, zero, or both #MISSING and zero values. Works within a designated range and retains all other members identical to the current member.

Syntax

```
@NEXTS (SKIPNONE | SKIPMISSING | SKIPZERO | SKIPBOTH mbrName [, n, rangeList])
```

Parameter Description

SKIPNONE Includes all cells specified in the *rangeList* operation, regardless of their content.

SKIPMISSING Ignores all #MISSING values in the *rangeList* operation.

SKIPZERO Ignores all 0 in the *rangeList* operation.

SKIPBOTH Ignores all #MISSING and 0 values in the *rangeList* operation.

mbrName Any valid single member name or member combination, or a function that returns a single member or member combination.

n Optional signed integer. Using a negative value for *n* has the same effect as using the matching positive value in [@PRIORS](#). If you do not specify *n*, then a default value of 1 is assumed, which returns the next prior member from the lowest level of the dimension set as Time in the database outline.

Parameter	Description
rangeList	Optional. A valid member, a comma-delimited list of member names, member set functions, and range functions. If <i>rangeList</i> is not specified, Essbase uses the level 0 members from the dimension set as Time.

Example

In this example, Next Cash for each month is derived by taking the Cash value for the following month and ignoring both #MISSING and zero values. Because *n* is not specified, the default is 1, which provides the next member in the range. Also, because *rangeList* is not specified, the level 0 members from the dimension set as Time are used (Jan, Feb, Mar, ...).

```
"Next Cash" = @NEXTS(SKIPBOTH, Cash);
```

The following report illustrates the above example:

	Jan	Feb	Mar	Apr	May	Jun
	===	===	===	===	===	===
Cash	1100	#MI	1000	1300	0	1400
Next Cash	1000	1000	1300	1400	1400	#MI

See Also

- [@NEXT](#)

@NEXTSIBLING

Returns the next sibling (the sibling to the immediate right) of the specified member. This function excludes the specified member. If the specified member is the last sibling, Essbase returns an empty string.

This function returns the next sibling as a string. To pass the @NEXTSIBLING function as a parameter of another function, where the function requires a list of members, you must wrap the output of @NEXTSIBLING with the @MEMBER function.

Syntax

```
@NEXTSIBLING (mbrName)
```

Parameter Description

mbrName Valid member name or member-name combination or a function that returns one member or member combination.

Example

All examples are from the Sample.Basic database.

```
@NEXTSIBLING("100-20")
```

Returns 100-30 (the next sibling of 100-20).

```
@NEXTSIBLING("200")
```

Returns 300 (the next sibling of 200). The @NEXTSIBLING and the @SHIFTSIBLING ("200", 1) function return the same results.

```
@MEMBER (@NEXTSIBLING ("100-20"))
```

Returns 100-30 (the next sibling of 100-20).

```
@CHILDREN (@MEMBER (@NEXTSIBLING ("East")))
```

Returns all children of West.

See Also

- [@PREVSIBLING](#)
- [@SHIFTSIBLING](#)

@NOTEQUAL

Returns a member set of member names that do not match the specified token name.

This function can be used on unique and duplicate-name outlines.

Syntax

```
@NOTEQUAL (tokenName, topMbrinHierarchy)
```

Parameter	Description
tokenName	Token string value, representing the name of a member, with which to compare to members in the outline, starting with member specified in <i>topMbrinHierarchy</i> . The specified token name must not be qualified for duplicate members.
topMbrinHierarchy	A fully qualified name of a member in the outline on which to base the member search. The specified member and its aliases, and all of its descendants, are included in the search. To search the entire outline, provide an empty string (" ") for this parameter. For example, <code>@NOTEQUAL ("300-30", " ")</code> .

Example

The following examples are based on the following duplicate-name outline:

```
Product
  100
    100-10
      100-10-10
    100-20
    100-30
  200
    200-10
    200-20
    200-30
  300
    300-10
    300-20
Diet
  100-10
    100-10-11
  200-10
  300-10
Bottle
```

200-10
300-20

@NOTEQUAL ("200-10", "Product")

Returns all of the members under the Product dimension, except for the members [Bottle]. [200-10], [Diet].[200-10], and [200].[200-10].

@NOTEQUAL ("200-10", "Diet")

Returns the members Diet, [Diet].[100-10], [Diet].[100-10].[100-10-10], and [Diet].[300-10].

See Also

- [@BETWEEN](#)
- [@EQUAL](#)
- [@EXPAND](#)
- [@LIKE](#)
- [@MBRCOMPARE](#)
- [@MBRPARENT](#)

@NPV

Calculates the Net Present Value of an investment based on the series of payments (negative values) and income (positive values).

Syntax

@NPV (*cashflowMbr*, *rateMbrConst*, *discountFlag* [, *rangeList*])

Parameter	Description
cashflowMbr	Member specification providing a series of numeric values.
rateMbrConst	Single member specification, variable name, or numeric expression, providing a constant value.
discountFlag	Single member specification, variable name, or numeric expression set to 0 or 1 to indicate whether the function should discount from the first period. 1 means do not discount from the first period.
rangeList	Optional. A valid member name, a comma-delimited list of member names, member set functions, and range functions from the dimension tagged as Time. If <i>rangeList</i> is not specified, Essbase uses the level 0 members from the dimension tagged as Time.

Notes

Financial functions never return a value; rather, they calculate a series of values internally based on the range specified.

Example

In this example, Value is calculated with the following formula:

```
Value = @NPV(Cash, Rate, 0, FY1990:FY1994, FY1995:FY2000);
```

This example produces the following report:

	FY1990	FY1991	FY1992	FY1993	FY1994	FY1995
	=====	=====	=====	=====	=====	=====
Cash	(1,000)	500	600	500	#MISSING	#MISSING
Rate	0	0	0	0	#MISSING	#MISSING
Value	296	296	296	296	296	296

See Also

- [@PTD](#)

@PARENT

Returns the parent of the current member being calculated in the specified dimension. If you specify the optional *mbrName*, that parent is combined with the specified member.

This member set function can be used as a parameter of another function, where that parameter is a member or list of members.

Syntax

```
@PARENT (dimName [, mbrName])
```

Parameter Description

dimName Single dimension name specification.

mbrName Optional. Any valid single member name or member combination, or a function that returns a single member or member combination, that is combined with the parent returned.

Notes

- You cannot use the @PARENT function in a FIX statement.
- You can use the @PARENT function on both the left and right sides of a formula. If you use this function on the left side of a formula in a calculation script, associate it with a member. For example:

```
Sales(@PARENT(Product) = 5);
```

- In some cases, the @PARENT function is equivalent to the @PARENTVAL function, except in terms of calculation performance. For example, the following two formulas are equivalent:

```
Sales = @PARENT(Profit);
Sales = @PARENTVAL(Profit);
```

In this case, using the latter formula results in better calculation performance. In general, use @PARENT as a member rather than as an implied value of a cell. For example:

```
Sales = @AVG(SKIPMISSING, @ISIBLINGS(@PARENT("100")));
```

- The time required for retrieval and calculation may be significantly longer if this function is in a formula attached to a member tagged as Dynamic Calc or Dynamic Calc and Store.
- If you are using the @PARENT function within @XREF: the @XREF function requires the @NAME function to be used around @PARENT. For example:

```
COGS=@XREF(Sample, @NAME(@PARENT(Product)), Sales);
```

Example

In the Sample Basic database:

```
@PARENT (Market, Sales)
```

returns Central->Sales, if the current member of Market being calculated is Colorado.

```
@PARENT (Measures)
```

returns Profit, if the current member of Measures being calculated is Margin.

See Also

- [@ANCEST](#)
- [@CHILDREN](#)
- [@ANCESTORS](#)
- [@DESCENDANTS](#)
- [@SIBLINGS](#)

@PARENTVAL

Returns the parent values of the member being calculated in the specified dimension.

Syntax

```
@PARENTVAL (dimName [, mbrName])
```

Parameter Description

dimName Single dimension name specification that defines the focus dimension of parent values.

mbrName Optional. Any valid single member name or member combination, or a function that returns a single member or member combination.

Example

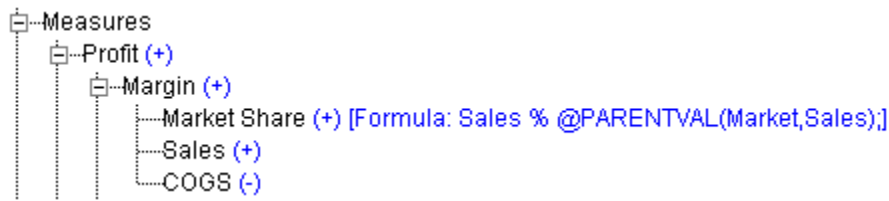
This example is based on the Sample Basic database. The formula calculates Market Share for each state by taking each state's Sales value as a percentage of Sales for East (its parent) as a whole. Market Share->East is calculated as East's percentage of its parent, Market.

```
"Market Share" = Sales % @PARENTVAL (Market, Sales);
```

This example produces the following report:

	Cola Sales =====	Actual Market Share =====	Jan Share
New York	678	37.42	
Massachusetts	494	27.26	
Florida	210	11.59	
Connecticut	310	17.11	
New Hampshire	120	6.62	
East	1812	37.29	
Market	4860	100	

Adding the "Market Share" member and formula to the outline would produce the same result as above.



See Also

- [@MDPARENTVAL](#)
- [@SPARENTVAL](#)
- [@ANCESTVAL](#)

@POWER

Returns the value of the specified member or expression raised to *power*.

Syntax

@POWER (*expression*, *power*)

Parameter Description

expression Single member specification, variable name, function, or other numeric expression.

power Single member specification, variable name, function, or other numeric expression.

Notes

- If *expression* is negative, and if *power* is not an integer, Essbase returns #MISSING.
- If the value calculated by @POWER is an infinite number, Essbase returns #MISSING.

Example

Usage	Return Value
@POWER(14, 3)	2744
@POWER(2, 8)	256

See Also

- [@FACTORIAL](#)

@PREVSIBLING

Returns the previous sibling (the sibling to the immediate left) of the specified member. This function excludes the specified member. If the specified member is the first sibling, Essbase returns an empty string.

This function returns the next sibling as a string. To pass the @PREVSIBLING function as a parameter of another function, where the function requires a list of members, you must wrap the output of @PREVSIBLING with the @MEMBER function.

Syntax

```
@PREVSIBLING (mbrName)
```

Parameter Description

mbrName Valid member name or member-name combination or a function that returns one member or member combination.

Example

All examples are from the Sample.Basic database.

```
@PREVSIBLING ("100-20")
```

Returns 100-10 (the previous sibling of 100-20). The @PREVSIBLING ("100-20") function and the @SHIFTSIBLING ("100-20", -1) function return the same results.

Returns 100 (the previous sibling of 200).

```
@PREVSIBLING ("100-10")
```

Returns an empty list as 100-10 does not have a previous sibling.

```
@CHILDREN (@MEMBER (@PREVSIBLING ("East")))
```

Returns an empty list as there is no previous sibling of East at the same level.

See Also

- [@NEXTSIBLING](#)
- [@SHIFTSIBLING](#)

@PRIOR

Returns the *n*th previous cell member from *mbrName* in *rangeList*. All other dimensions assume the same members as the current member. @PRIOR works only within the designated range, and with level 0 members.

Syntax

```
@PRIOR (mbrName [, n, rangeList])
```

Parameter Description

mbrName Any valid single member name or member combination, or a function that returns a single member or member combination.

n Optional signed integer. Using a negative value for *n* has the same effect as using the matching positive value in the @NEXT function. If you do not specify *n*, then a default value of 1 is assumed, which returns the next prior member from the lowest level of the dimension tagged as Time in the database outline.

Parameter Description

rangeList Optional. A valid member name, a comma-delimited list of member names, member set functions, and range functions. If *rangeList* is not specified, Essbase uses the level 0 members from the dimension tagged as Time.

Example

In this example, Prev Inventory for each month is derived by taking the Inventory value from the previous month. Since *n* is not specified, the default is 1, which provides the next prior member in the range. Since *rangeList* is not specified, the level 0 members from the dimension tagged as Time are used (Jan, Feb, Mar, ...).

```
"Prev Inventory" = @PRIOR(Inventory);
```

This example produces the following report:

	Jan	Feb	Mar	Apr	May	Jun
	===	===	===	===	===	===
Inventory	1100	1200	1000	1300	1300	1400
Prev Inventory	#MI	1100	1200	1000	1300	1300

See Also

- [@NEXT](#)
- [@SHIFT](#)
- [@SHIFTMINUS](#)
- [@SHIFTPLUS](#)

@PRIORS

Returns the *n*th previous cell member from *mbrName* in the *rangeList*. @PRIORS provides options to skip #MISSING, zero, or both #MISSING and zero values. All other dimensions assume the same members as the current member. @PRIORS works within the designated range.

Syntax

```
@PRIORS(SKIPNONE | SKIPMISSING | SKIPZERO | SKIPBOTH mbrName[,n, rangeList])
```

Parameter Description

SKIPNONE Includes all cells specified in the *rangeList* operation regardless of their content.

SKIPMISSING Ignores all #MISSING values in the *rangeList* operation.

SKIPZERO Ignores all zero values in the *rangeList* operation.

SKIPBOTH Ignores all #MISSING and zero values in the *rangeList*.

mbrName Any valid single member name or member combination, or a function that returns a single member or member combination.

Parameter	Description
n	Optional signed integer. Using a negative value for <i>n</i> has the same effect as using the matching positive value in the @NEXTS function. If you do not specify <i>n</i> , then a default value of 1 is assumed, which returns the next prior member from the lowest level of the dimension set as Time in the database outline.
rangeList	Optional. A valid member, a comma-delimited list of member names, member set functions, and range functions. If <i>rangeList</i> is not specified, Essbase uses the level 0 members from the dimension set as Time.

Example

In this example, Prev Inventory for each month is derived by taking the Inventory value from the previous month and ignoring #MISSING and zero values. Because *n* is not specified, the default is 1, which provides the next prior member in the range. Also, because rangeList is not specified, the level 0 members from the dimension are set as Time used as (Jan, Feb, Mar, ...).

```
"Prev Inventory" = @PRIORS (SKIPBOTH, Inventory) ;
```

The following report illustrates this example:

	Jan	Feb	Mar	Apr	May	Jun
	===	===	===	===	===	===
Inventory	1100	#MI	1000	1300	0	1400
Prev Inventory	#MI	1100	1100	1000	1300	1300

See Also

- [@PRIOR](#)

@PTD

Calculates the period-to-date values of members in the dimension tagged as Time. By default, data is summed unless Accounts are tagged as "First" or "Last".

Syntax

```
@PTD (timePeriodList)
```

Parameter	Description
-----------	-------------

timePeriodList	Range of members from the dimension tagged as Time.
----------------	-----------------------------------------------------

Notes

- Financial functions never return a value; rather, they calculate a series of values internally based on the range specified.
- You can use the @PTD function only if the outline contains a dimension tagged as Accounts.

Example

In this example, assume that the Year dimension in the Sample Basic database outline contains two additional members, YTD and QTD. Using a calculation script, the YTD and QTD members are calculated as follows:

YTD = @PTD(Jan:May) ;
 QTD = @PTD(Apr:May) ;

In this example Opening Inventory is tagged with a time balance of First, and Ending Inventory is tagged with a time balance of Last.

This example produces the following report:

	Product	Market	Scenario
	Sales	Opening Inventory	Ending Inventory
	=====	=====	=====
Jan	31538	117405	116434
Feb	32069	116434	115558
Mar	32213	115558	119143
Qtr1	95820	117405	119143
Apr	32917	119143	125883
May	33674	125883	136145
Jun	35088	136145	143458
Qtr2	101679	119143	143458
QTD	66591	245026	262028
YTD	162411	362431	381171

See Also

- [@NPV](#)

@RANGE

Returns a member list that crosses the specified member from one dimension (*mbrName*) with the specified member range from another dimension (*rangeList*). @RANGE can be combined with non-range functions, such as @AVG, which replaces an existing range function, such as @AVGRANGE.

Syntax

@RANGE (*mbrName* [, *rangeList*])

Parameter Description

- mbrName** Any valid single member name or member combination, or a function that returns a single member or member combination.
- rangeList** Optional. A valid member name, a comma-delimited list of member names, member set functions, and range functions. If *rangeList* is not specified, Essbase uses the level 0 members from the dimension tagged as Time.

Notes

Calculator function @RANGE and the cross-dimensional operator (->) cannot be used inside a FIX/ENDFIX statement.

Example

Example 1

The following example is based on the Sample Basic database. The @RANGE function is used with the @AVG function to determine the average sales for Colas in the West.

```
FIX(Sales)
West=@AVG(SKIPBOTH,@RANGE(Sales,@CHILDREN(West)));
ENDFIX
```

Since the calculation script fixes on Sales, only the Sales value for West are the average of the values for western states; COGS values for West are the sum of the western states. This example produces the following report:

	Colas			
	Sales		COGS	
	Actual		Actual	
	Qtr3	Qtr4	Qtr3	Qtr4
	====	====	====	====
California	3401	2767	2070	1701
Oregon	932	1051	382	434
Washington	1426	1203	590	498
Utah	1168	1294	520	575
Nevada	496	440	222	197
West	1484.6	1351	3784	3405

Example 2

The following example is based on the Sample Basic database. Assume that the Measures dimension contains an additional member, Prod Count. The @RANGE function is used with the @COUNT function to calculate the count of all products for which a data value exists:

```
"Prod Count" = @COUNT(SKIPMISSING,@RANGE(Sales,@CHILDREN(Product)));
```

This example produces the following report. Since SKIPMISSING is specified in the formula, the #MI value for Sales->Diet Drinks is not counted as a data value:

	Jan	New York	Actual
	Sales	Prod Count	
	=====	=====	
Colas	678	#MI	
Root Beer	551	#MI	
Cream Soda	663	#MI	
Fruit Soda	587	#MI	
Diet Drinks	#MI	#MI	
Product	2479	4	

See Also

- [@LIST](#)
- [@MERGE](#)
- [@REMOVE](#)

@RANK

Returns the rank of the specified members or the specified value among the values in the specified data set. The rank of a value is equivalent to its position (its rank) in the sorted data set.

Syntax

```
@RANK (SKIPNONE | SKIPMISSING | SKIPZERO | SKIPBOTH, value, expList)
```

Parameter	Description
SKIPNONE	Includes all cells specified in <i>expList</i> , regardless of their content, during calculation of the rank.
SKIPMISSING	Excludes all #MISSING values from <i>expList</i> during calculation of the rank.
SKIPZERO	Excludes all zero (0) values from <i>expList</i> during calculation of the rank.
SKIPBOTH	Excludes all zero (0) values and #MISSING values from <i>expList</i> during calculation of the rank.
value	(1) The member or member combination for which the rank is calculated, or (2) a constant value for which the rank is calculated.
expList	Comma-delimited list of member specifications, variable names, functions, or numeric expressions. <i>expList</i> provides a list of numeric values across which the rank is calculated.

Notes

- After SKIP processing, the @RANK function sorts the data set in descending order (for example, 15341, 9650, 6556, 4255, 1989). The rank of a value identifies its position in the sorted data set (for example, 15341 is ranked 1; 1989 is ranked 5)
- An input value of #MISSING returns #MISSING. #MISSING is also returned if, after SKIP processing, there are no values to compare.
- The @RANK function assigns the same rank to duplicate values; however, the presence of duplicate values affects the rank numbers. For example, if a list of values contains [2,2,4,5], Essbase first sorts the list [5,4,2,2] and then ranks: [5] has a rank of 1, [4] has a rank of 2, and [2] has a rank of 3. In this case, no value has a rank of 4.
- If *value* is a constant value and that value is not included in *expList*, Essbase inserts the constant value in the list and then ranks it accordingly. For example, if a list of values contains [2,4,6,13], and you want to rank a value of [3] in this list, Essbase:
 1. Sorts the list in descending order [13,6,4,2]
 2. Inserts [3] in the list [13,6,4,3,2]
 3. Ranks [3] in the list: in this case, [3] has a rank of 4.
- When you use @RANK in a calculation script, use it within a FIX statement. Although using FIX is not required, it may improve calculation performance.
- When you use @RANK across a large range in a sparse dimension, you may need to increase the size of the calculator cache. For more information on the calculator cache, see the *Oracle Essbase Database Administrator's Guide*.

Example

The following example is based on the Sample Basic database. Assume that the Measures dimension contains an additional member, Sales Rank. Essbase ranks the sales values for a set of products:

```
"Sales Rank" = @RANK(SKIPBOTH, Sales,
@RANGE(Sales, @LEVMBRS(Product, 1)));
```

This example produces the following report. Since SKIPBOTH is specified in the formula, the #MI value for Sales->Diet Drinks is not included in the ranked list:

	New York	Actual	Jan
	Sales	Sales	Rank
	=====	=====	
Colas	678		1
Root Beer	551		4
Cream Soda	663		2
Fruit Soda	587		3
Diet Drinks	#MI		#MI

@RDESCENDANTS

Returns all descendants of the specified member, or those down to the specified generation or level, including shared members. This function excludes the specified member.

You can use this member set function as a parameter of another function, where that parameter is a list of members.

In the absence of shared members, @RDESCENDANTS and @DESCENDANTS return the same result.

Syntax

```
@RDESCENDANTS (mbrName [, genLevNum| genLevName])
```

Parameter	Description
-----------	-------------

mbrName	Any valid single member name or member combination, or a function that returns a single member or member combination
---------	----------------------------------------------------------------------------------------------------------------------

genLevNum	Optional. An integer value that defines the absolute generation or level number down to which to select the members. A positive integer defines a generation number. A value of 0 or a negative integer defines a level number.
-----------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

genLevName	Optional. Level name or generation name down to which to select the members.
------------	------------------------------------------------------------------------------

Notes

- The order of members in the result list is important to consider when you use the @RDESCENDANTS member set function with certain forecasting and statistical functions. Essbase generates the list of members in the following sequence: If a shared member is encountered, the above steps are repeated on the member being shared.
 1. The specified member
 2. The nearest descendant of the member

3. The next nearest descendant of the member, and so on.

- You can use [@IRDESCENDANTS](#) to include the specified member in the member list.

Example

Example 1

Assume a variation of the Sample Basic database such that the Product dimension includes the following members:

```
Product
  100
    100-10
    100-20
    100-30
  200
    200-10
    200-20
    200-30
    200-40
Diet
  100 (Shared Member)
  200 (Shared Member)
```

Diet has two children "100" and "200". The members "100" and "200" are shared members.

```
@RDESCENDANTS(Diet)
```

returns the members: 100, 100-10, 100-20, 100-30, 200, 200-10, 200-20, 200-30, 200-40 (in that order).

Example 2

```
@RDESCENDANTS(Profit)
```

returns Margin, Sales, COGS, Total Expenses, Marketing, Payroll, and Misc (in that order) and is identical to [@DESCENDANTS\(Profit\)](#).

See Also

- [@DESCENDANTS](#)
- [@IRDESCENDANTS](#)
- [@IDESCENDANTS](#)
- [@ISDESC](#)
- [@ANCESTORS](#)
- [@CHILDREN](#)
- [@SIBLINGS](#)

@RELATIVE

Returns all members at the specified generation or level that are above or below the specified member in the database outline.

Syntax

`@RELATIVE (mbrName, genLevNum | genLevName)`

Parameter	Description
-----------	-------------

mbrName	Any valid single member name or member combination, or a function that returns a single member or member combination.
---------	-----------------------------------------------------------------------------------------------------------------------

genLevNum	An integer value that defines the number of a generation or level. A positive integer defines a generation number. A value of 0 or a negative integer defines a level number.
-----------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

genLevName	Generation or level name specification.
------------	-----------------------------------------

Notes

This function returns all members at the specified generation or level that are above or below the specified member in the database outline.

Essbase sorts the generated list of members in ascending order. Using Sample Basic as an example, `@RELATIVE(200, 0)`, returns 200-10, 200-20, 200-30, 200-40 (in that order). This order is important to consider when you use the `@RELATIVE` member set function with certain forecasting and statistical functions.

Example

In the Sample Basic database:

```
@RELATIVE(Qtr1, 3)
@RELATIVE(Qtr1, 0)
```

both return the three members that are at generation 3 (or level 0) and that are below Qtr1 in the Sample Basic outline: Jan, Feb, and Mar (in that order).

```
@RELATIVE(Profit, -1)
```

returns the two members that are at level 1 and that are below Profit: Margin and Total Expenses (in that order).

@REMAINDER

Returns the remainder value of *expression*.

Syntax

`@REMAINDER (expression)`

Parameter	Description
-----------	-------------

expression	Single member specification, variable name, or other numeric expression.
------------	--------------------------------------------------------------------------

Example

```
Margin = @REMAINDER("Margin %");
```

This example produces the following report:

	Product			Market		
	Jan	Feb	Mar	Jan	Feb	Mar
	===	===	===	===	===	===
Scenario	55.10	55.39	55.27	0.10	0.39	0.27

See Also

- [@TRUNCATE](#)

@REMOVE

Removes values or members in one list from another list.

Syntax

```
@REMOVE (list1, list2)
```

Parameter Description

list1 A list of member specifications, from which the members specified in *list2* are removed.

list2 A list of member specifications to be removed from *list1*.

Example

Example 1

In the Sample Basic database,

```
@REMOVE (@CHILDREN (East), @LIST ("New York", Connecticut))
```

returns Massachusetts, Florida, New Hampshire.

Example 2

The following example is based on the Sample Basic database. Assume that the Market dimension contains an additional member, Non-West.

A special analysis requires a sum of the actual sales values of a particular product family for non-western states. In this example, the @REMOVE function is used with the @SUMRANGE function to perform this analysis. The @LIST function is used to group the last two arguments of the @REMOVE function (the children of West plus two additional members, Texas and New Mexico).

```
FIX (Sales)
"Non-West" = @SUMRANGE (Sales, @REMOVE (@LEVMBRS (Market, 0),
@LIST (@CHILDREN (West), Texas, "New Mexico")));
ENDFIX
```

This example produces the following report:

	Jan	Colas	Actual
		Sales	
		=====	
Non-West		5114	

New York	678
Massachusetts	494
Florida	410
Connecticut	310
New Hampshire	213
East	2105
California	941
Oregon	450
Washington	320
Utah	490
Nevada	138
West	2339
Texas	642
Oklahoma	180
Louisiana	166
New Mexico	219
South	1207
Illinois	579
Ohio	430
Wisconsin	490
Missouri	360
Iowa	161
Colorado	643
Central	2663

See Also

- [@LIST](#)
- [@MERGE](#)
- [@RANGE](#)

@RETURN

Exits the calculation immediately under specified logical conditions. You can use the IF... ELSEIF calculation command block to specify the logical error conditions, and use the @RETURN function to exit the calculation with customized error messages and levels.

Syntax

```
@RETURN ( "ErrorMessage", [ , INFO | ERROR | WARNING ] )
```

Parameter	Description
-----------	-------------

ErrorMessage	An error message string, or any expression that returns a string.
--------------	-------------------------------------------------------------------

Parameter	Description
INFO ERROR WARNING	<p>An error message priority setting, where INFO, ERROR, and WARNING are priority levels:</p> <ul style="list-style-type: none"> • INFO—The message indicated in the <i>ErrorMessage</i> string is sent back to the client and the application log as an informational type message. This is the default. • ERROR—The message indicated in the <i>ErrorMessage</i> string is sent back to the client and the application log as an error type message. • WARNING—The message indicated in the <i>ErrorMessage</i> string is sent back to the client and the application log as a warning type message.

Notes

- The calculation script will stop executing when this function is called.
- This function can only be used in calculation scripts; it cannot be used in member formulas.

Example

The following example stops the calculation and returns a custom warning message if maximum values specified in the IF statement are empty:

```
FIX("Actual")
. "Profit"(
    IF( ("Marketing" < 0) OR ("Payroll" < 0) OR ("Misc" < 0) )
        @RETURN( @CONCATENATE(
            @CONCATENATE("The violation of data integrity : Market [",
@NAME(@CURRMBR("Market")),
            "] has a negative expenses. Calculations are interrupted")
            , WARNING);
        ELSE
            "Profit" = ("Margin" - "Total Expenses")*0.9;
        ENDF
    )
ENDFIX
```

@ROUND

Rounds *expression* to *numDigits*.

Syntax

```
@ROUND (expression, numDigits)
```

Parameter Description

expression Single member specification, variable name, or other numeric expression.

Parameter Description

numDigits Single member specification, variable name, or other numeric expression that provides an integer value. If *numDigits* is 0 or a positive number, *expression* is rounded to the number of decimal places specified by *numDigits*. If *numDigits* is a negative value, *expression* is rounded to the nearest 10 to the power of the absolute value of *numDigits*. For example:

```
@ROUND 1234, -2) = 1200
```

The default value for *numDigits* is 0.

Example

The following example is based on the Sample Basic database:

```
SET UPDATECALC OFF;  
Profit = @ROUND("Profit_%", 1);
```

This example produces the following report:

	Market			Product		
	Jan	Feb	Mar	Jan	Feb	Mar
Scenario	21.37	19.09	18.46	21.4	19.1	18.5

See Also

- [@ABS](#)
- [@INT](#)
- [@TRUNCATE](#)
- [@REMAINDER](#)

@RSIBLINGS

Returns the right siblings of the specified member.

Syntax

```
@RSIBLINGS (mbrName)
```

Parameter Description

mbrName Any valid single member name or member combination, or a function that returns a single member or member combination.

Notes

This function returns all of the right siblings of the specified member. Right siblings are children that share the same parent as the member and that follow the member in the database outline. This function excludes the specified member.

This member set function can be used as a parameter of another function, where that parameter is a list of members.

Essbase sorts the right siblings in ascending order. Using Sample Basic as an example, if you specify 200-10 for *mbrName*, Essbase returns 200-20, 200-30, 200-40 (in that order). This order

is important to consider when you use the @RSIBLINGS member set function with certain forecasting and statistical functions.

Example

In the Sample Basic database:

```
@RSIBLINGS(Florida)
```

returns Connecticut and New Hampshire (in that order). These members appear below Florida in the Sample Basic outline.

```
@RSIBLINGS(Sales)
```

returns COGS because this member appears below Sales in the Sample Basic outline.

See Also

- [@RSIBLINGS](#)
- [@LSIBLINGS](#)
- [@NEXTSIBLING](#)
- [@PREVSIBLING](#)
- [@SHIFTSIBLING](#)

@SANCESTVAL

Returns ancestor-level data based on the shared ancestor value of the current member being calculated.

Syntax

```
@SANCESTVAL (rootMbr, genLevNum [, mbrName])
```

Parameter	Description
-----------	-------------

<i>rootMbr</i>	Defines a member that is used to search for the nearest occurrence of an ancestor of a shared member.
<i>genLevNum</i>	Integer value that defines the absolute generation or level number from which the ancestor values are to be returned. A positive integer defines a generation reference. A negative number or value of 0 defines a level reference.
<i>mbrName</i>	Optional. Any valid single member name or member combination, or a function that returns a single member or member combination, for which the ancestor values are to be returned.

Notes

- You cannot use the @SANCESTVAL function in a FIX statement.
- The time required for retrieval and calculation may be significantly longer if this function is in a formula attached to a member tagged as Dynamic Calc or Dynamic Calc and Store.

Example

Marketing expenses are captured at the Product Category levels in a product planning application. The Product categories are defined as ancestors that contain shared members as

children. The Marketing Expense data must be allocated down to each Product code based on Sales contribution.

The following Product hierarchy is defined:

```

Product
  100
    100-10
    100-20
  200
    200-10
    200-20
Diet ~
  100-10 SHARED
  200-10 SHARED
Caffeine Free ~
  100-20 SHARED
  200-20 SHARED

```

	Sales	Marketing
	=====	=====
100-10	300	0
100-20	200	0
100	500	0
200-10	100	0
200-30	400	0
200	900	0
100-10	300	0
200-10	100	0
Diet	400	50
100-20	200	0
200-30	400	0
Caffeine Free	600	40

The Marketing Expense value is allocated down to each Product code with the following formula:

```
Marketing = (Sales / @SANCESTVAL(Product, 2, Sales)) * @SANCESTVAL(Product, 2, Marketing);
```

which produces the following result:

	Sales	Marketing
	=====	=====
100-10	300	37.5
100-20	200	13.3
100	500	#MI
200-10	100	12.5
200-30	400	26.7
200	900	#MI
100-10	300	37.5
200-10	100	12.5
Diet	400	50
100-20	200	13.3
200-30	400	26.7
Caffeine Free	600	40

The Marketing expenses can then be reconsolidated across Products and Markets.

See Also

- [@ANCESTVAL](#)
- [@MDANCESTVAL](#)
- [@SPARENTVAL](#)

@SHARE

Checks each member from *rangeList* to see if it has a shared member and returns a list of the shared members it has found.

Syntax

```
@SHARE (rangeList)
```

Parameter Description

rangeList A comma-delimited list of members, functions that return members, and ranges of members. All the members in *rangeList* must be from the same dimension.

Notes

Other member-set functions return actual members, not the shared members. You can use @SHARE within the *memberList*, *rangeList*, *explist* or *list* parameters of other functions to provide shared members instead.

Example

The following examples are based on Sample Basic.

To remove all shared members from the Product dimension:

```
@REMOVE (@DESCENDANT (Product) , @SHARE (@DESCENDANT ( (Product) ) )
```

To remove a specific member from the Product dimension, you can use @SHARE specifying the shared member to be removed:

```
@REMOVE (@DESCENDANT (Product) , @SHARE ("100-20" ) )
```

See Also

- [@REMOVE](#)

@SHIFT

Returns either the prior or next *n*th cell value in the sequence *rangeList* from *mbrName*, retaining all other members identical to the current member.

The direction of @SHIFT is wholly based on *n*, with positive *n* values producing an effect equivalent to @NEXT and negative values of *n* producing an equivalent effect to @PRIOR.

Syntax

```
@SHIFT (mbrName [, n, rangeList])
```

Parameter Description

mbrName	Any valid single member name or member combination, or a function that returns a single member or member combination.
n	Optional signed integer. Using a negative value for <i>n</i> has the same effect as using a positive value in the @PRIOR function. <i>n</i> must be a numeric value, not a reference, such as a member name.
rangeList	Optional. A valid member name, a comma-delimited list of member names, member set functions, and range functions. If <i>rangeList</i> is not specified, Essbase uses the level 0 members from the dimension tagged as Time.

Notes

@SHIFT is provided as a more appropriate, self-documenting name than @NEXT or @PRIOR when the value for *n* is a variable and may change from positive to negative, depending on the database state when the call occurs (that is, when the usage is likely to be NEXT and/or PRIOR).

Example

In this example, Prev Asset for each month is derived by taking the Asset value from the previous month because -1 is specified as the *n* parameter. Next Avl Asset for each month is derived by taking the Asset value from two months following the current month because 2 is specified as the *n* parameter. Since *rangeList* is not specified for either formula, the level 0 members from the dimension tagged as Time are used.

```
"Prev Asset" = @SHIFT(Asset, -1);  
"Next Avl Asset" = @SHIFT(Asset, 2);
```

This example produces the following report:

	Jan	Feb	Mar	Apr	May	Jun
	===	===	===	===	===	===
Asset	100	110	105	120	115	125
Prev Asset	#MI	100	110	105	120	115
Next Avl Asset	105	120	115	125	#MI	#MI

See Also

- [@MDSHIFT](#)
- [@NEXT](#)
- [@PRIOR](#)
- [@SHIFTPLUS](#)
- [@SHIFTMINUS](#)

@SHIFTMINUS

Can be used in place of the @SHIFT() function, the @PRIOR() function, or the @NEXT() function to improve performance if the formula meets the following criteria:

- The formula is being executed in CELL mode.
- The formula has one of the following patterns:
$$X = Y - @SHIFT(mbrName [, n, rangeList])$$

or:

```
X = Y - @PRIOR(mbrName [,n, rangeList])
```

or:

```
X = Y - @NEXT(mbrName [,n, rangeList])
```

If these criteria are met, consider rewriting your formula using @SHIFTMINUS() instead. @SHIFTMINUS() runs the formula in block mode, improving performance.

Syntax

```
@SHIFTMINUS (mbrName1, mbrName2 [,n, rangeList])
```

Parameter	Description
mbrName1mbrName2	Any valid single member name or member combination, or a function that returns a single member or member combination.
n	Optional signed integer. <i>n</i> must be a numeric value, not a reference, such as a member name. If you are using @SHIFTPLUS to replace the @NEXT function, use 1 as the value for <i>n</i> . If you are using @SHIFTPLUS to replace the @PRIOR function, use -1 as the value for <i>n</i> . Default value is +1.
rangeList	Optional. A valid member name, a comma-delimited list of member names, member set functions, and range functions. If <i>rangeList</i> is not specified, Essbase uses the level 0 members from the dimension tagged as time.

Example

The following example shows a formula using @SHIFT().

```
Sales = Loss - @SHIFT(Sales, 1);
```

Here is the formula using @SHIFTMINUS() to improve performance:

```
@SHIFTMINUS (Loss, Sales, 1)
```

See Also

- [@SHIFT](#)
- [@SHIFTPLUS](#)
- [@PRIOR](#)
- [@NEXT](#)

@SHIFTPLUS

Can be used in place of the @SHIFT() function, the @PRIOR() function, or the @NEXT() function to improve performance if the formula meets the following criteria:

- The formula is being executed in CELL mode.
- The formula has one of the following patterns:

```
X = Y + @SHIFT(mbrName [,n, rangeList])
```

or:

```
X = Y + @PRIOR(mbrName [,n, rangeList])
```

or:

```
X = Y + @NEXT(mbrName [,n, rangeList])
```

If these criteria are met, consider rewriting your formula using @SHIFTPLUS() instead. @SHIFTPLUS() runs the formula in block mode, improving performance.

Syntax

```
@SHIFTPLUS (mbrName1, mbrName2 [,n, rangeList])
```

Parameter	Description
mbrName1mbrName2	Any valid single member name or member combination, or a function that returns a single member or member combination.
n	Optional signed integer. <i>n</i> must be a numeric value, not a reference, such as a member name. If you are using @SHIFTPLUS to replace the @NEXT function, use 1 as the value for <i>n</i> . If you are using @SHIFTPLUS to replace the @PRIOR function, use -1 as the value for <i>n</i> . Default value is +1.
rangeList	Optional. A valid member name, a comma-delimited list of member names, member set functions, and range functions. If <i>rangeList</i> is not specified, Essbase uses the level 0 members from the dimension tagged as time.

Example

The following example shows a formula using @SHIFT().

```
Sales = Loss + @SHIFT(Sales, 1);
```

Here is the formula using @SHIFTPLUS() to improve performance:

```
@SHIFTPLUS (Loss, Sales, 1);
```

See Also

- [@SHIFT](#)
- [@SHIFTMINUS](#)
- [@PRIOR](#)
- [@NEXT](#)

@SHIFTSIBLING

Returns the specified member or the *n*th sibling of the member. @SHIFTSIBLING traverses members that are at the same level and of the same parent. If the specified relative position moves beyond the first or last sibling, Essbase returns an empty string.

This function returns the next sibling as a string. To pass the @SHIFTSIBLING function as a parameter of another function, where the function requires a list of members, you must wrap the output of @SHIFTSIBLING with the @MEMBER function.

Syntax

```
@SHIFTSIBLING (mbrName [,relativePosition])
```

Parameter	Description
mbrName	Valid member name or member-name combination or a function that returns one member or member combination.
relativePosition	Optional. The integer that defines the position relative to the specified member. Valid values: <ul style="list-style-type: none"> ● 0 (Default) Returns the specified member. ● < 0 (negative integer): Returns the previous sibling. ● > 0 (positive integer): Returns the next sibling.

Example

All examples are from the Sample.Basic database.

```
@SHIFTSIBLING("100-20", 0)
```

Returns 100-20 (the specified member).

```
@SHIFTSIBLING("200", 1)
```

Returns 300 (the next sibling of 200). The @SHIFTSIBLING("200", 1) function and the @NEXTSIBLING("200") function return the same results.

Returns 400 (the second-next sibling of 200).

```
@SHIFTSIBLING("100-20", -1)
```

Returns 100-10 (the previous sibling of 100-20). The @SHIFTSIBLING("100-20", -1) function and the @PREVSIBLING("100-20") function return the same results.

```
@SHIFTSIBLING("100-10", 9)
```

Returns an empty string, as 100-10 does not have a ninth sibling.

```
@CHILDREN(@MEMBER(@SHIFTSIBLING("East")))
```

Returns all children of East. Because no shift position is specified, the default shift position is 0, which means the current member.

See Also

- [@PREVSIBLING](#)
- [@NEXTSIBLING](#)

@SIBLINGS

Returns all siblings of the specified member.

Syntax

```
@SIBLINGS (mbrName)
```

Parameter Description

mbrName Any valid single member name or member combination, or a function that returns a single member or member combination.

Notes

This function returns all siblings of the specified member. This function excludes the specified member.

This function can be used as a parameter of another function, where that parameter is a list of members.

Essbase sorts the generated list of members as follows:

1. Left siblings of the member (siblings appearing above the member in the database outline) in descending order
2. Right siblings of the member (siblings appearing below the member in the database outline) in ascending order

Using Sample Basic as an example, if you specify 200-30 for *mbrName*, Essbase returns 200-20, 200-10, 200-40 (in that order). This order is important to consider when you use the @SIBLINGS member set function with certain forecasting and statistical functions.

Example

In the Sample Basic database:

```
@SIBLINGS (Washington)
```

Returns Oregon, California, Utah, and Nevada (in that order).

```
@SIBLINGS (East)
```

Returns West, South, and Central (in that order).

See Also

- [@ISIBLINGS](#)
- [@ISISIBLING](#)
- [@ISSIBLING](#)
- [@LSIBLINGS](#)
- [@RSIBLINGS](#)
- [@SHIFTSIBLING](#)
- [@NEXTSIBLING](#)
- [@PREVSIBLING](#)

@SLN

Calculates the periodic amount that an asset in the current period may be depreciated, calculated across a range of periods. The depreciation method used is straight-line depreciation:

$$\text{cost} - \text{salvage value} / \text{life}$$

The SLN method assumes that the asset depreciates by the same amount each period.

More than one asset may be depreciated over the range. The value is depreciated from its entry period to the last period in the range. The resulting value represents the sum of all the per-period depreciation values of each asset being depreciated.

Syntax

@SLN (*costMbr*, *salvageMbrConst*, *lifeMbrConst* [, *rangeList*])

Parameter	Description
costMbr	Single member specification representing an input asset for the current period.
salvageMbrConst	Single member specification, variable name, or numeric expression, providing a constant numeric value. This value represents the value of the asset in the current period at the end of the useful life of the asset.
lifeMbrConst	Single member specification, variable name, or numeric expression representing the useful life of the asset.
rangeList	Optional. A valid member name, a comma-delimited list of member names, member set functions, and range functions from the dimension tagged as Time. <i>rangeList</i> represents the range over which the function accepts input and returns depreciation values. If <i>rangeList</i> is not specified, Essbase uses the level 0 members from the dimension tagged as Time.

Notes

Financial functions never return a value; rather, they calculate a series of values internally based on the range specified.

Example

In this example, the depreciation for each year is calculated by taking into account the initial asset (Asset), the salvage value of the asset (Residual), and the life of the asset (Life).

```
"SLN Dep" = @SLN(Asset,Residual,Life,FY1991:FY1995);
```

This example produces the following report:

	FY1991	FY1992	FY1993	FY1994	FY1995	FY1996
	=====	=====	=====	=====	=====	=====
Asset	9,000	0	1,000	0	0	0
Residual	750.00	0.00	0.00	0.00	0	0
Life	5.00	#MI	5.00	0.00	0.00	0
SLN Dep	1650	1650	1850	1850	1850	0

See Also

- [@DECLINE](#)
- [@SYD](#)

@SPARENTVAL

Returns parent-level data based on the shared parent value of the current member being calculated.

Syntax

@SPARENTVAL (*RootMbr* [, *mbrName*])

Parameter Description

RootMbr Defines a member that is used to search for the nearest occurrence of a parent of a shared member.

mbrName Optional. Any valid single member name or member combination, or a function that returns a single member or member combination, from which the parent values are returned.

Notes

- You cannot use the @SPARENTVAL function in a FIX statement.
- The time required for retrieval and calculation may be significantly longer if this function is in a formula attached to a member tagged as Dynamic Calc or Dynamic Calc and Store.

Example

Marketing expenses are captured at the Product Category levels in a product planning application. The Product categories are defined as parents that contain shared members as children. The Marketing Expense data must be allocated down to each Product code based on Sales contribution.

The following Product hierarchy is defined:

```

Product
100
    100-10
    100-20
200
    200-10
    200-20
Diet ~
    100-10 SHARED
    200-10 SHARED Caffeine Free ~
    100-20 SHARED
    200-20 SHARED

```

	Sales	Marketing
	=====	=====
100-10	300	0
100-20	200	0
100	500	0
200-10	100	0
200-30	400	0
200	900	0
100-10	300	0
200-10	100	0
Diet	400	50
100-20	200	0
200-30	400	0
Caffeine Free	600	40

The Marketing Expense value is allocated down to each Product code with the following formula:

$$\text{Marketing} = (\text{Sales} / @\text{SPARENTVAL}(\text{Product}, \text{Sales})) * @\text{SPARENTVAL}(\text{Product}, \text{Marketing});$$

which produces the following result:

Sales	Marketing
=====	=====

100-10	300	37.5
100-20	200	13.3
100	500	#Missing
200-10	100	12.5
200-30	400	26.7
200	900	#Missing
100-10	300	37.5
200-10	100	12.5
Diet	400	#Missing
100-20	200	13.3
200-30	400	26.7
Caffeine Free	600	#Missing

The Marketing expenses can then be reconsolidated across Products and Markets.

See Also

- [@PARENTVAL](#)
- [@MDPARENTVAL](#)
- [@SANCESTVAL](#)

@SPLINE

Applies a smoothing spline to a set of data points. A spline is a mathematical curve that smoothes or interpolates data.

Syntax

```
@SPLINE (YmbrName [, s [, XmbrName [, XrangeList]])
```

Parameter Description

YmbrName	A valid single member name that contains the dependent variable values used (when crossed with <i>rangeList</i>) to construct the spline.
s	Optional. A zero (0) or positive value that determines the smoothness parameter. The default value is 1.0.
XmbrName	Optional. A valid single member name that contains the independent variable values used (when crossed with <i>rangeList</i>) to construct the spline. The default independent variable values are 0,1,2,3, and so on.
XrangeList	Optional. A valid member name, a comma-delimited list of member names, cross dimension members, or a member set function or range function (including @XRANGE) that returns a list of members from the same dimension. If <i>XrangeList</i> is not specified, Essbase uses the level 0 members from the dimension tagged as time.

Notes

- *XrangeList* must contain at least two values.
- If *XrangeList* contains gaps in the data (for example: Jan, Feb, Mar, Jun, Jul), be sure to specify *XmbrName* (for example: 0,1,2,5,6) so that correct results are returned.
- The @SPLINE function skips #MISSING values in *YmbrName* and *XmbrName*; in the result, Essbase replaces the #MISSING values of *YmbrName* with the spline values.
- The @SPLINE function calculates a smoothing cubic spline for ($n > 0$).

- Setting the smoothness parameter (s) to 0 produces an interpolating spline, that is, a spline that fits the initial data exactly. Increasing s results in a smoother spline but a less exact approximation of the initial data.
- The @SPLINE function can be used with the @TREND function to forecast future values that are based on the values smoothed with @SPLINE.
- If you use an Essbase member set function to generate a member list for the *XrangeList* parameter (for example, @SIBLINGS), to ensure correct results, consider the order in which Essbase sorts the generated member list. For more information, see the *Oracle Essbase Technical Reference* topic for the member set function you are using.
- When you use @SPLINE in a calculation script, use it within a FIX statement. Although using FIX is not required, it may improve calculation performance.
- When you use @SPLINE across a large range in a sparse dimension, you may need to increase the size of the calculator cache. For more information on the calculator cache, see the *Oracle Essbase Database Administrator's Guide*.
- View the [Algorithm](#) for the smoothing spline.

Algorithm

$$(x_i, y_i), \quad i = 0, 1, \dots, N$$

A function $S(x)$ defined on grid $X = \{x_i\}$ is called a *smoothing cubic spline function* if the function

- 1) is a cubic polynomial

$$S(x) = S_i(x) = a_0^{(i)} + a_1^{(i)}(x - x_i) + a_2^{(i)}(x - x_i)^2 + a_3^{(i)}(x - x_i)^3$$

on each partial segment $[x_i, x_{i+1}]$, $i = 0, 1, \dots, N-1$,

- 2) has the continuous second derivatives on segment $[x_0, x_N]$, that is, the function is of class $C^2[x_0, x_N]$,
- 3) minimizes the functional

$$J(f) = s \int_{x_N}^{x_0} (f''(x))^2 dx + \sum_{i=0}^N (f(x_i) - y_i)^2,$$

where y_i are given numbers and $s \geq 0$, where s is the smoothness parameter, and

4) satisfies the boundary condition:

$$S''(x_0) = 0, \quad S''(x_N) = 0$$

In each segment $[x_i, x_{i+1}]$, $i = 0, 1, \dots, N-1$, the smoothing spline function is sought in the following modified form:

$$S(x) = S_i(x) = z_i(1-t) + z_{i+1}t - \frac{h_i^2}{6}t(1-t)[(2-t)n_i + (1+t)n_{i+1}]. \quad (*)$$

where

$$h_i = x_{i+1} - x_i, \quad t = \frac{x - x_i}{h_i}, \quad (**)$$

and numbers z_i and n_i , $i = 0, 1, \dots, N$ are a solution of a linear algebraic system.

The numbers n_i are solutions to the system:

$$a_0 n_0 + b_0 n_1 + c_0 n_2 = g_0,$$

$$b_0 n_0 + a_1 n_1 + b_1 n_2 + c_1 n_3 = g_1,$$

$$c_{i-2} n_{i-2} + b_{i-1} n_{i-1} + a_i n_i + b_i n_{i+1} + c_i n_{i+2} = g_i, \quad i = 2, 3, \dots, N-2,$$

$$c_{N-3} n_{N-3} + b_{N-2} n_{N-2} + a_{N-1} n_{N-1} + b_{N-1} n_N = g_{N-1},$$

$$c_{N-2} n_{N-2} + b_{N-1} n_{N-1} + a_N n_N = g_N,$$

where

$$a_i = \frac{1}{3}(h_{i-1} + h_i) + \frac{1}{h_{i-1}^2} s + \left(\frac{1}{h_{i-1}} + \frac{1}{h_i} \right)^2 s + \frac{1}{h_i^2} s,$$

$$i = 1, 2, \dots, N-1,$$

$$b_i = \frac{1}{6} h_i - \frac{s}{h_i} \left[\left(\frac{1}{h_{i-1}} + \frac{1}{h_i} \right) + \left(\frac{1}{h_i} + \frac{1}{h_{i+1}} \right) \right],$$

$$i = 1, 2, \dots, N - 2,$$

$$c_i = \frac{s}{h_i h_{i+1}}, \quad i = 1, 2, \dots, N - 3$$

$$g_i = \frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}}, \quad i = 1, 2, \dots, N - 1$$

The end conditions are:

$$a_0 = 1, \quad b_0 = 0, \quad c_0 = 0, \quad g_0 = 0,$$

$$a_N = 1, \quad b_{N-1} = 0, \quad c_{N-2} = 0, \quad g_N = 0.$$

When numbers n_i are found, the magnitudes z_i are easily determined by formulas

$$z_i = y_i - s D_i, \quad i = 0, 1, 2, \dots, N,$$

where

$$D_0 = \frac{1}{h_0}(n_1 - n_0), \quad D_N = -\frac{1}{h_{N-1}}(n_N - n_{N-1}),$$

$$D_i = \frac{1}{h_i}(n_{i+1} - n_i) - \frac{1}{h_{i-1}}(n_i - n_{i-1}), \quad i = 1, 2, \dots, N - 1.$$

And now given any x , use (*) and (***) from above to calculate $S(x)$.

Example

The following example is based on the Sample Basic database. Assume that the Measures dimension contains an additional member, Sales Spline. The formula calculates the spline of Sales values for Jan through Jun, based on a smoothness parameter of 2.

```
"Sales Spline" = @SPLINE(Sales,2,,Jan:Jun);
```

This example produces the following report:

	Colas Sales	Actual Sales	New York Sales Spline
	=====		=====
Jan	645		632.8941564
Feb	675		675.8247101
Mar	712		724.7394598
Apr	756		784.2860765
May	890		852.4398456
Jun	912		919.8157517

See Also

- [@TREND](#)

@STDEV

Calculates the standard deviation of the specified data set (*expList*). The calculation is based upon a sample of a population. Standard deviation is a measure of how widely values are dispersed from their mean (average).

This function assumes that *expList* represents a sample of a population. If you want *expList* to represent the entire population, use [@STDEVP](#). For large samples, the functions return similar values.

@STDEV is calculated using the "nonbiased" or "n-1" method.

@STDEV uses the following formula:

$$\sqrt{\frac{n \sum x^2 - (\sum x)^2}{n(n-1)}}$$

Syntax

@STDEV (SKIPNONE | SKIPMISSING | SKIPZERO | SKIPBOTH, *expList*)

Parameter	Description
SKIPNONE	Includes all cells specified in <i>expList</i> , regardless of their content, during calculation of the standard deviation.
SKIPMISSING	Excludes all #MISSING values from <i>expList</i> during calculation of the standard deviation.
SKIPZERO	Excludes all zero (0) values from <i>expList</i> during calculation of the standard deviation.
SKIPBOTH	Excludes all zero (0) values and #MISSING values from <i>expList</i> during calculation of the standard deviation.
<i>expList</i>	Comma-delimited list of member specifications, variable names, functions, or numeric expressions. <i>expList</i> provides a list of numeric values across which the standard deviation is calculated.

Notes

The @STDEV function replaces the @STDDEV function. The only difference between the functions is the SKIP parameter in the @STDEV function. Although the old @STDDEV function is supported for migration purposes, you can no longer select it in the Calculation Script Editor or Formula Editor.

Example

The following example is based on the Sample Basic database. Assume that the Measures dimension contains an additional member, Std Deviation. This example calculates the standard deviation (based on a sample of a population) of the sales values for all products and uses the [@RANGE](#) function to generate *expList*.

```
FIX (Product)
"Std Deviation" = @STDEV(SKIPBOTH,@RANGE(Sales,@CHILDREN(Product)));
ENDFIX
```

This example produces the following report:

		Jan	New York
		Actual	Budget
		=====	=====
Sales	Colas	678	640
	Root Beer	551	530
	Cream Soda	663	510
	Fruit Soda	587	620
	Diet Drinks	#MI	#MI
	Product	2479	2300
Std Deviation	Product	60.73	64.55

See Also

- [@RANGE](#)
- [@STDEVP](#)
- [@STDEV RANGE](#)

@STDEVP

Calculates the standard deviation of the specified data set (*expList*).

Syntax

```
@STDEVP (SKIPNONE | SKIPMISSING | SKIPZERO | SKIPBOTH, expList)
```

Parameter	Description
SKIPNONE	Includes all cells specified in <i>expList</i> , regardless of their content, during calculation of the standard deviation.
SKIPMISSING	Excludes all #MISSING values from <i>expList</i> during calculation of the standard deviation.
SKIPZERO	Excludes all zero (0) values from <i>expList</i> during calculation of the standard deviation.
SKIPBOTH	Excludes all zero (0) values and #MISSING values from <i>expList</i> during calculation of the standard deviation.
expList	Comma-delimited list of member specifications, variable names, functions, or numeric expressions. <i>expList</i> provides a list of numeric values across which the standard deviation is calculated.

Notes

@STDEVP calculates the standard deviation of the specified data set (*expList*). The calculation is based upon the entire population. Standard deviation is a measure of how widely values are dispersed from their mean (average).

This function assumes that *expList* represents the entire population. If you want *expList* to represent a sample of a population, use @STDEV. For large samples, the functions return similar values.

@STDEVP is calculated using the "biased" or "n" method.

@STDEVP uses the following formula:

$$\sqrt{\frac{n \sum x^2 - (\sum x)^2}{n^2}}$$

Example

The following example is based on the Sample Basic database. Assume that the Measures dimension contains an additional member, Std Deviation. This example calculates the standard deviation (based on the entire population) of the sales values for all products and uses the [@RANGE](#) function to generate *expList*.

```
FIX (Product)
"Std Deviation" = @STDEVP(SKIPBOTH, @RANGE(Sales, @CHILDREN(Product)));
ENDFIX
```

This example produces the following report:

		Jan	New York
		Actual	Budget
		=====	=====
Sales	Colas	678	640
	Root Beer	551	530
	Cream Soda	663	510
	Fruit Soda	587	620
	Diet Drinks	#MI	#MI
	Product	2479	2300
Std Deviation	Product	52.59	55.90

See Also

- [@RANGE](#)
- [@STDEV](#)
- [@STDEV RANGE](#)

@STDEV RANGE

Calculates the standard deviation of all values of the specified member (*mbrName*) across the specified data set (*XrangeList*). The calculation is based upon a sample of a population. Standard deviation is a measure of how widely values are dispersed from their mean (average).

This function is calculated using the "unbiased" or "n-1" method. See [@STDEV](#) for the formula used.

Syntax

```
@STDEV RANGE (SKIPNONE | SKIPMISSING | SKIPZERO | SKIPBOTH, mbrName [, XrangeList])
```

Parameter	Description
SKIPNONE	Includes all cells specified in <i>expList</i> , regardless of their content, during calculation of the standard deviation.
SKIPMISSING	Excludes all #MISSING values from <i>expList</i> during calculation of the standard deviation.
SKIPZERO	Excludes all zero (0) values from <i>expList</i> during calculation of the standard deviation.
SKIPBOTH	Excludes all zero (0) values and #MISSING values from <i>expList</i> during calculation of the standard deviation.
mbrName	Any valid single member name or member combination, or a function that returns a single member or member combination.
XrangeList	Optional. A valid member name, a comma-delimited list of member names, cross dimension members, or a member set function or range function (including @XRANGE) that returns a list of members from the same dimension. If <i>XrangeList</i> is not specified, Essbase uses the level 0 members from the dimension tagged as time.

Notes

The @STDEV RANGE function replaces the @STDDEV RANGE function. The only difference between the functions is the SKIP parameter in the @STDEV RANGE function. Although the old @STDDEV RANGE function is supported for migration purposes, you can no longer select it in the Calculation Script Editor or Formula Editor.

Example

The following example is based on the Sample Basic database. Assume that the Measures dimension contains an additional member, Std Deviation. This example calculates the standard deviation (based on a sample of a population) of the sales values for all products.

```
FIX (Product)
"Std Deviation" = @STDEV RANGE(SKIPBOTH, Sales, @CHILDREN(Product));
ENDFIX
```

This example produces the following report:

		Jan	New York
		Actual	Budget
		=====	=====
Sales	Colas	678	640
	Root Beer	551	530
	Cream Soda	663	510
	Fruit Soda	587	620
	Diet Drinks	#MI	#MI
	Product	2479	2300
Std Deviation	Product	60.73	64.55

See Also

- [@STDEV](#)
- [@STDEV P](#)

@SUBSTRING

Returns the requested string of characters from an existing source string. The source string can be a text string or a member name, or it can result from a specified function that returns a text string or a single member name.

Syntax

```
@SUBSTRING (String, StartPosition [, EndPosition])
```

Parameter	Description
-----------	-------------

String	A string or a function that returns a string or a single member name (For example, @ATTRIBUTESVAL, @CONCATENATE, and @NAME return strings.)
--------	---------------------------------------------------------------------------------------------------------------------------------------------

StartPosition	Beginning character position within <i>String</i> to include in the substring. An integer greater than or equal to 0, where 0 corresponds to the first character in <i>String</i> , 1 corresponds to the second character, and so on.
---------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

EndPosition	Optional. An integer greater than or equal to 1, where 1 corresponds to the first character in <i>String</i> , 2 corresponds to the second character, and so on. If <i>EndPosition</i> is not specified or is less than <i>StartPosition</i> , Essbase returns all remaining characters from the source string. Note that this is a different numbering scheme that the start position uses.
-------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Example

The following examples are based on the Sample Basic database:

Function Statement	Result
@SUBSTRING ("100-10",1)	"00-10"
@SUBSTRING ("200-21",0,2)	"20"
@SUBSTRING (@Name(@Parent(Jan)),3) (The parent of Jan is Qtr1.)	"1"

See Also

- [@CONCATENATE](#)
- [@MEMBER](#)

@SUM

Returns the summation of all the values in *expList*.

Syntax

```
@SUM (expList)
```

Parameter	Description
-----------	-------------

expList	Comma-delimited list of member specifications, variable names, or numeric expressions, all of which provide numeric values.
---------	-----------------------------------------------------------------------------------------------------------------------------

Example

In the Sample Basic database:

```
FIX("Total Expenses")
West=@SUM(West, East);
ENDFIX
```

Since the calculation script fixes on Total Expenses, the value for Total Expenses->West is equal to the sum of the value for East and the values for the states making up the West. For Sales, West and East are simply the sum of the states making up each region (that is, Sales->West is not equal to the sum of East and West). This example produces the following report:

	Product	Qtr1	Actual
	Sales	Total	Expenses
	=====	=====	=====
New York	7705		2068
Massachusetts	3660		892
Florida	4132		1313
Connecticut	3472		1087
New Hampshire	1652		801
East	20621		6161
California	11056		2742
Oregon	5058		1587
Washington	4835		1621
Utah	4209		1544
Nevada	6516		2193
West	31674		15848

See Also

- [@SUMRANGE](#)

@SUMRANGE

Returns the summation of all the values of the specified member (*mbrName*) across the specified range (*XrangeList*).

Syntax

```
@SUMRANGE (mbrName [, XrangeList])
```

Parameter Description

mbrName Any valid single member name, or a function that returns a single member.

Note: Member name cannot be a cross-dimensional member combination.

XrangeList Optional. A valid member name, a comma-delimited list of member names, cross dimension members, or a member set function or range function (including @XRANGE) that returns a list of members from the same dimension. If *XrangeList* is not specified, Essbase uses the level 0 members from the dimension tagged as time.

Example

The following example is based on the Sample Basic database. Assume that the Year dimension contains an additional member, Partial Year. The formula for Partial Year sums the values for New York across the range of Jan through Jun. The calculation script fixes on Sales, so this formula is applied only to Sales values.

```
FIX(Sales)
"Partial Year"=@SUMRANGE("New York",Jan:Jun);
ENDFIX
```

This example produces the following report:

Actual	New York	Colas
	Sales	
	=====	
Jan	678	
Feb	645	
Mar	675	
Apr	712	
May	756	
Jun	890	
Partial Year	4356	

See Also

- [@SUM](#)

@SYD

Calculates the periodic amount (usually annual) that an asset in the current period may be depreciated, across a range of periods. The depreciation method used is sum of the year's digits.

The SYD method assumes that depreciation amounts are higher at the earlier stages of the asset's life. Thus, *rangeList* can be used to specify a period to calculate.

More than one asset may be depreciated over the range. The value is depreciated from its entry period to the last period in the range. The resulting value represents the sum of all per-period depreciation values of each asset.

Syntax

```
@SYD (costMbr, salvageMbrConst, lifeMbrConst [, rangeList])
```

Parameter	Description
<i>costMbr</i>	Single member specification representing an input asset for the current period.
<i>salvageMbrConst</i>	Single member specification, variable name, or numeric expression, providing a constant numeric value. This value is the value of the asset in the current period after the useful life of the asset.
<i>lifeMbrConst</i>	Single member specification, variable name, or numeric expression representing the useful life of the asset.

Parameter	Description
rangeList	Optional. A valid member name, a comma-delimited list of member names, member set functions, and range functions from the dimension tagged as Time. <i>rangeList</i> represents the range over which the function accepts input and returns depreciation values. If <i>rangeList</i> is not specified, Essbase uses the level 0 members from the dimension tagged as Time.

Notes

Financial functions never return a value; rather, they calculate a series of values internally based on the range specified.

Example

In this example, the depreciation for each year is calculated by taking into account the initial asset (Asset), the salvage value of the asset (Residual), and the life of the asset (Life).

```
"SYD Dep"=@SYD(Asset,Residual,Life,FY1999:FY2002,FY2003);
```

This example produces the following report:

	FY1999	FY2000	FY2001	FY2002	FY2003
	=====	=====	=====	=====	=====
Asset	9,000	0	1,000	0	0
Residual	750.00	0.00	0.00	0.00	0
Life	5.00	#MISSING	3.00	0.00	0.00
SYD Dep	2750	2200	2150	1433	717

See Also

- [@DECLINE](#)
- [@SLN](#)

@TODATE

Converts date strings to numbers that can be used in calculation formulas. @TODATE converts date strings into the number of seconds elapsed since midnight, January 1, 1970.

Syntax

```
@TODATE (formatString, dateString)
```

Parameter	Description
-----------	-------------

formatString	The format of the date string, either "mm-dd-yyyy" or "dd-mm-yyyy" (must be in lower case).
--------------	---------------------------------------------------------------------------------------------

dateString	The date string.
------------	------------------

Notes

- If you specify a date that is earlier than 01-01-1970, this function returns an error.
- The latest date supported by this function is 12-31-2037.

Example

The following example is based on the Sample Basic database.

```
Marketing
(IF (@ATTRIBUTEVAL("Intro Date") >
    @TODATE("mm-dd-yyyy", "06-30-1996"))
Marketing - (Marketing * .1);
ENDIF);
```

This formula searches for members with an Intro Date attribute member that is later than 6-30-96 and decreases Marketing for those members by 10 percent. In order to process the formula, Essbase converts the date strings to numbers before it calculates.

This example produces the following report:

	Actual	Jan Marketing	Massachusetts
Intro Date_12-10-1996	200-30	9	
	200-40	9	
Intro Date_10-01-1996	400-10	9	
	400-20	9	
Intro Date_07-26-1996	200-20	9	
Intro Date_06-26-1996	300-10	9	
	300-20	9	
	300-30	9	
Intro Date_04-01-1996	100-20	10	
	100-30	10	
Intro Date_03-25-1996	100-10	10	
Intro Date_09-27-1995	200-10	10	

See Also

- [@ATTRIBUTE](#)
- [@ATTRIBUTEVAL](#)
- [@WITHATTR](#)

@TODATEEX

Returns the numeric date value from input date-string according to the date-format specified. The date returned is the number of seconds elapsed since midnight, January 1, 1970.

If the date or the date format strings are invalid, an error is returned.

Syntax

```
@TODATEEX(date_format_string, string)
```

Parameter	Description
date_format_string	<p>One of the following literal strings (excluding ordered-list numbers and parenthetical examples) indicating a supported date format.</p> <ol style="list-style-type: none"> 1. "mon dd yyyy" (Example: mon = Aug) 2. "Month dd yyyy" (Example: Month = August) 3. "mm/dd/yy" 4. "mm/dd/yyyy" 5. "yy.mm.dd" 6. "dd/mm/yy" 7. "dd.mm.yy" 8. "dd-mm-yy" 9. "dd Month yy" 10. "dd mon yy" 11. "Month dd, yy" 12. "mon dd, yy" 13. "mm-dd-yy" 14. "yy/mm/dd" 15. "yymmdd" 16. "dd Month yyyy" 17. "dd mon yyyy" 18. "yyyy-mm-dd" 19. "yyyy/mm/dd" 20. Long format (Example: WeekDay, Mon dd, yyyy) 21. Short format (Example: m/d/yy)

Parameter	Description
string	<p>A date string following the rules of <i>internal-date-format</i>. The following examples correspond to the above listed internal date formats.</p> <ol style="list-style-type: none"> 1. Jan 15 2006 2. January 15 2006 3. 01/15/06 4. 01/15/2006 5. 06.01.06 6. 15/01/06 7. 15.01.06 8. 15-01-06 9. 15 January 06 10. 15 Jan 06 11. January 15 06 12. Jan 15 06 13. 01-15-06 14. 06/01/15 15. 060115 16. 15 January 2006 17. 15 Jan 2006 18. 2006-01-15 19. 2006/01/15 20. Sunday, January 15, 2006 21. 1/8/06 (m/d/yy)

Notes

- This function is case-sensitive. For example, using `apr` instead of `Apr` returns an error.
- Using extra whitespace not included in the internal format strings returns an error.
- Trailing characters after the date format has been satisfied are ignored. If you erroneously use a date string of `06/20/2006` with date format `mm/dd/yy`, the trailing `06` is ignored and the date is interpreted as June 20, 2020.
- Long Format (Weekday, Mon dd, yyyy) is not verified for a day-of-week match to the given date.

For example: For date string `Sunday, March 13, 2007` with date format `Long Format`, the input date string is parsed correctly for `March 13, 2007`, although `March 13, 2007` does not fall on Sunday.

- If you specify a date that is earlier than `01-01-1970`, this function returns an error.
- The latest date supported by this function is `12-31-2037`.
- When the `yy` format is used, this function interprets years in the range 1970 to 2029.

See Also

- [@DATEDIFF](#)
- [@DATEPART](#)
- [@DATEROLL](#)
- [@FORMATDATE](#)
- [@TODAY](#)

@TODAY

Returns a number representing the current date on the Essbase computer. The number is the number of seconds elapsed since midnight, January 1, 1970.

Syntax

@TODAY ()

Notes

The *date* returned can be used as input to other functions listed in the See Also section.

See Also

- [@DATEDIFF](#)
- [@DATEPART](#)
- [@DATEROLL](#)
- [@FORMATDATE](#)
- [@TODATEEX](#)

@TREND

Calculates future values based on curve-fitting to historical values. The @TREND procedure considers a number of observations; constructs a mathematical model of the process based on these observations (that is, fits a curve); and predicts values for a future observation. You can use weights to assign credibility coefficients to particular observations, report errors of the curve fitting, choose the forecasting method to be used (for example, linear regression), and specify certain data filters.

Syntax

```
@TREND (Ylist, [Xlist], [weightList], [errorList], [XforecastList], YforecastList,  
method[, method parameters] [, Xfilter1 [, parameters]] [, XfilterN [, parameters]] [,  
Yfilter1 [, parameters]] [, YfilterN [, parameters]])
```

Parameter	Description
-----------	-------------

Ylist	An expression list that contains known observations; for example, sales figures over a period of time.
-------	--------------------------------------------------------------------------------------------------------

Xlist	Optional. An expression list that contains underlying variable values. For example, for each sales figure in <i>Ylist</i> , <i>Xlist</i> may contain a value for associated time periods. If you do not specify <i>Xlist</i> , the default variable values are 1,2,3, and so on, up to the number of values in <i>Ylist</i> .
-------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Parameter	Description
weightList	Optional. An expression list that contains weights for the data points in <i>Ylist</i> , for the linear regression method only. If values in <i>weightList</i> are #MISSING, the default is 1. Weights for methods other than linear regression are ignored. Negative weights are replaced with their absolute values.
errorList	Optional. Member list that represents the differences between the data points in <i>Ylist</i> and the data points on the line or curve (as specified for <i>method</i>).
XforecastList	Optional. Expression list that contains the underlying variable values for which the forecasting is sought. If you do not specify <i>XforecastList</i> , the values are assumed to be as follows: {(last value in <i>Xlist</i> + 1), (last value in <i>Xlist</i> + 2), ...} up to (last value in <i>Xlist</i> + the number of values in <i>YforecastList</i>) If you forecast consecutively from where <i>Ylist</i> stops, you do not need to specify <i>XforecastList</i> . If you want to move the forecasting period forward, specify the new period with <i>XforecastList</i> .
YforecastList	A member list into which the forecast values are placed.
method	A choice among LR (linear regression), SES (single exponential smoothing), DES (double exponential smoothing), and TES (triple exponential smoothing). Method parameters must be numeric values, not member names. Method parameters may be any of the following: <ul style="list-style-type: none"> ● <i>LR</i>[,<i>t</i>]: standard linear regression with possible weights assigned to each data point and an optional seasonal adjustment period [<i>t</i>], where [<i>t</i>] is the length of the period. In general, the weights are equal to 1 by default. You might want to increase the weight if the corresponding observation is important, or decrease the weight if the corresponding observation is an outlier or is unreliable. ● <i>SES</i>[,<i>c</i>]: single exponential smoothing with parameter <i>c</i> (default <i>c</i>=0.2). This method uses its own weight system, using the single parameter <i>c</i>. Increasing this parameter gives more weight to early observations than to later ones. ● <i>DES</i>[[,<i>c1</i>],<i>c2</i>]: double exponential smoothing (Holt's method) with optional parameters <i>c1</i>, <i>c2</i> (default <i>c1</i>=0.2, <i>c2</i>=0.3). This is a two-parameter weight system and a linear subsequent approximation scheme. The first parameter controls weight distribution for the intercept; the second parameter controls weight distribution for the slope of the line fit. ● <i>TES</i>[[[[,<i>T</i>],<i>c1</i>],<i>c2</i>],<i>c3</i>]: triple exponential smoothing (Holt-Winters method) with optional parameters <i>c1</i>, <i>c2</i>, <i>c3</i>, <i>T</i> (default <i>c1</i>=0.2, <i>c2</i>=0.05, <i>c3</i>=0.1, <i>T</i>=1). This is a three-parameter weight system and a linear model with a multiplicative seasonal component.
Xfilter1 ... XfilterN	Optional. Use one or more of the following filter methods to scale <i>Xlist</i> : <ul style="list-style-type: none"> ● <i>XLOG</i>[,<i>c</i>]: logarithmic change with shift <i>c</i> ($x' = \log(x+c)$) (default <i>c</i>=1) ● <i>XEXP</i>[,<i>c</i>]: exponential change with shift <i>c</i> ($x' = \exp(x+c)$) (default <i>c</i>=0). ● <i>XPOW</i>[,<i>c</i>]: power change with power <i>c</i> ($x' = x^c$) (default <i>c</i>=2).
Yfilter1 ... YfilterN	Optional. Use one or more of the following filter methods to scale <i>Ylist</i> : <ul style="list-style-type: none"> ● <i>YLOG</i>[,<i>c</i>]: logarithmic change with shift <i>c</i> ($y' = \log(y+c)$) (default <i>c</i>=1) ● <i>YEXP</i>[,<i>c</i>]: exponential change with shift <i>c</i> ($y' = \exp(y+c)$) (default <i>c</i>=0). ● <i>YPOW</i>[,<i>c</i>]: power change with power <i>c</i> ($y' = y^c$) (default <i>c</i>=2).

Notes

- The @TREND function can be used only in calculation scripts, not in outline formulas.
- In a calculation script, you must associate the @TREND formula with a member.
- *Ylist*, *Xlist*, *weightList*, and *errorList* should contain the same number of values.
- *XforecastList* and *YforecastList* should contain the same number of values.

- The *method* and *filter* parameters must be numbers only; functions and member names are not allowed.
- @TREND ignores #MISSING values during calculation of the trend.
- When you use the LR method with seasonal adjustments or when you use the TES method, Essbase places strict requirements on the input data. With these methods, input data cannot contain #MISSING values. Also, if you specify *Xlist*, the data must be equidistant, with the interval (step) being a whole fraction of the period, T (for example, T/5, T/2). The *XforecastList* parameters should also contain multiples of the interval.
- For another example using the @TREND function with more options used, see the *Oracle Essbase Database Administrator's Guide*.
- If you use a member set function to generate a member list for this function, (for example, @SIBLINGS), to ensure correct results, consider the order in which Essbase sorts the generated member list. For more information, see the *Oracle Essbase Technical Reference* topic for the member set function you are using.
- The following algorithms are used to calculate @TREND:

Algorithm for Linear Regression

Ylist y_1, y_2, \dots, y_x

Xlist x_1, x_2, \dots, x_x

weightList w_1, w_2, \dots, w_x

Linear Regression (LR)

(if w_i is #MISSING or the whole *weightList* is missing as an argument, $w_i = 1$)

$$S = \sum_{i=1}^K (w_i)^2 \quad S_x = \sum_{i=1}^K x_i (w_i)^2 \quad S_y = \sum_{i=1}^K y_i (w_i)^2$$

$$S_{xx} = \sum_{i=1}^K (x_i)^2 (w_i)^2 \quad S_{xy} = \sum_{i=1}^K (x_i y_i) (w_i)^2$$

$$\Delta = SS_{xx} - (S_x)^2$$

$$a = \frac{S_x S_y - S_x S_{xy}}{\Delta}$$

$$b = \frac{SS_{xy} - S_x S_y}{\Delta}$$

the equation of the line is:

$$\text{line} = Y_{LR}(x) = a + bx$$

Algorithm for Double Exponential Smoothing (DES)

$$Ylist \quad y_1, y_2, \dots, y_K$$

$$Xlist \quad x_1, x_2, \dots, x_K$$

$c_1 = .2, \quad c_2 = .3$ default, or else they are input into the trend

find S_1, S_2, \dots, S_K

b_1, b_2, \dots, b_K

$$S_1 = y_1$$

$$b_1 = \frac{(y_2 - y_1)}{(x_2 - x_1)}$$

$$S_{i+1} = a_i * (S_i + b_i(x_{i+1} - x_i)) + (1 - a_i) * (y_{i+1})$$

$$b_{i+1} = d_i * b_i + (1 - d_i) * \left[\frac{(S_{i+1} - S_i)}{x_{i+1} - x_i} \right]$$

$$\text{where } a_i = (1 - c_1)^{x_{i+1} - x_i}$$

$$d_i = (1 - c_2)^{x_{i+1} - x_i}$$

$$y_1 \quad y_2 \quad y_3 \quad y_4 \quad y_5 \quad y_6$$

In linear regressions, the intervals between x values must be the same.
The value of that interval is Δ . In this case, $\Delta = 1$.

Step 1, Centered moving average of y's, where $n = 3$ (moving centered average with 3 members at a time)

$$\begin{array}{cccccc}
 y_1 & y_2 & y_3 & y_4 & y_5 & y_6 \\
 \underbrace{\hspace{1.5cm}} & & & & & \\
 * & \frac{y_1 + y_2 + y_3}{3} & \frac{y_2 + y_3 + y_4}{3} & \frac{y_3 + y_4 + y_5}{3} & \frac{y_4 + y_5 + y_6}{3} & * \\
 & = \tilde{y}_2 & = \tilde{y}_3 & = \tilde{y}_4 & = \tilde{y}_5 & \leftarrow \boxed{\text{centered moving average}}
 \end{array}$$

Algorithm for Linear Regression with Seasonal Adjustment

$$Ylist \quad y_1, y_2, \dots, y_K$$

$$Xlist \quad x_1, x_2, \dots, x_K$$

$$weightList \quad w_1, w_2, \dots, w_K$$

$$@TREND(Ylist, \dots, LR, t)$$

Linear regression with seasonal adjustment example:

There are 6 data points and a seasonal adjustment parameter, $t=3$

Input data:

$$x_1 = 1 \quad x_2 = 2 \quad x_3 = 3 \quad x_4 = 4 \quad x_5 = 5 \quad x_6 = 6$$

$$y_1 \quad y_2 \quad y_3 \quad y_4 \quad y_5 \quad y_6$$

In linear regressions with seasonal adjustments, the intervals between x values must be the same. Δ is equal to that interval. In this case, $\Delta = 1$.

Step 1, Centered moving average of y's, where $n = 3$ (moving centered average with 3 members at a time)

$$\begin{array}{cccccc}
 y_1 & y_2 & y_3 & y_4 & y_5 & y_6 \\
 \underbrace{\hspace{1.5cm}} & & & & & \\
 * & \frac{y_1 + y_2 + y_3}{3} & \frac{y_2 + y_3 + y_4}{3} & \frac{y_3 + y_4 + y_5}{3} & \frac{y_4 + y_5 + y_6}{3} & * \\
 & = \tilde{y}_2 & = \tilde{y}_3 & = \tilde{y}_4 & = \tilde{y}_5 & \leftarrow \boxed{\text{centered moving average}}
 \end{array}$$

Step 2, Subtract \tilde{y} 's from y 's:

$$\begin{array}{cccc}
 y_2 & y_3 & y_4 & y_5 \\
 - \tilde{y}_2 & \tilde{y}_3 & \tilde{y}_4 & \tilde{y}_5 \\
 \hline
 \hat{y}_2 & \hat{y}_3 & \hat{y}_4 & \hat{y}_5 \leftarrow \text{difference}
 \end{array}$$

Step 3, Arrange \hat{y} 's into $n(n = 3)$ columns to derive P 's and average values along columns:

$$\begin{array}{ccc}
 * & \hat{y}_2 & \hat{y}_3 \\
 \hat{y}_4 & \hat{y}_5 & * \\
 \hline
 \frac{\hat{y}_4}{1} & \frac{\hat{y}_2 + \hat{y}_5}{2} & \frac{\hat{y}_3}{1} \\
 = P_0 & = P_1 & = P_2 \leftarrow \text{adjustment list}
 \end{array}$$

Step 4, Subtract P 's from original $Ylist$:

$$\begin{array}{cccccc} y_1 & y_2 & y_3 & y_4 & y_5 & y_6 \\ P_0 & P_1 & P_2 & P_0 & P_1 & P_2 \\ \hline y'_1 & y'_2 & y'_3 & y'_4 & y'_5 & y'_6 \end{array}$$

Step 5, Linear Regression (LR) with

$$\begin{array}{cccccc} x_1 = 1 & x_2 = 2 & x_3 = 3 & x_4 = 4 & x_5 = 5 & x_6 = 6 \\ y'_1 & y'_2 & y'_3 & y'_4 & y'_5 & y'_6 \end{array}$$

as shown in **Linear Regression (LR) section**, deriving a, b such that $y = bx + a$ is the trending line.

Step 6, To get future trend value for x :

$$\begin{aligned} x: \quad Y_{forecast} &= b * x + a + P_i, \quad \text{where } P_i: i = \frac{(x - x_1) \bmod t}{\Delta} \\ &= \frac{(x - 1) \bmod 3}{1} \\ &= (x - 1) \bmod 3 \end{aligned}$$

Algorithm for Single Exponential Smoothing (SES)

$$Ylist \quad y_1, y_2, \dots, y_K$$

$$Xlist \quad x_1, x_2, \dots, x_K$$

$c = .2$ default, or else c is input into the trend

find S_1, S_2, \dots, S_K :

$$S_1 = y_1$$

$$S_{i+1} = a_i * S_i + (1 - a_i) y_i \quad \text{for } i = 1, \dots, K - 1$$

$$\text{then } Y_{\text{forecast}}(x) = a * S_K + (1 - a) * y_K$$

$$\text{where } a_i = (1 - c)^{x_{i+1} - x_i}$$
$$a = (1 - c)^{x - x_K}$$

Note: When $Xlist$ is missing, $x_{i+1} - x_i = 1$ and the corresponding exponents and coefficients disappear.

Algorithm for Triple Exponential Smoothing (TES)

$$Ylist \quad y_1, y_2, \dots, y_K$$

$$Xlist \quad x_1, x_2, \dots, x_K$$

TES with period T (if T is not given, it is assumed to be $T = 1$)

$x_1, x_2, \dots, x_K, \quad y_1, y_2, \dots, y_K$ are input to TES, x is forecast value.

$$a_i = (1 - c)^{x_{i+1} - x_i} \quad d_i = (1 - d)^{x_{i+1} - x_i} \quad e_i = (1 - e)^{x_{i+1} - x_i}$$

Note: When $Xlist$ is missing, the exponents disappear.

Default

$$c = .2$$
$$d = .05$$
$$e = .1$$

Step 1,

$$S_1 = y_1$$
$$b_1 = \frac{y_2 - y_1}{x_2 - x_1}$$
$$I_1 = 1$$

Step 2, For $i = 1, \dots, T - 1$

$$S_{i+1} = a_i * (S_i + b_i (x_{i+1} - x_i)) + (1 - a_i) * \frac{y_i}{I_i}$$

$$I_{i+1} = \frac{y_i}{S_i}$$

$$b_{i+1} = d_i b_i + (1 - d_i) \frac{S_{i+1} - S_i}{x_{i+1} - x_i}$$

Step 3, For $i = T, \dots, K$

$$S_{i+1} = a_i * (S_i + b_i (x_{i+1} - x_i)) + (1 - a_i) \frac{y_{i+1}}{I_{i+1-T}}$$

$$I_{i+1} = e_i I_{i+1-T} + (1 - e_i) \frac{y_{i+1}}{S_{i+1}}$$

$$b_{i+1} = d_i b_i + (1 - d_i) \frac{S_{i+1} - S_i}{x_{i+1} - x_i}$$

Forecast for x is $(S_x + b_x (x - x_x)) * (I_j)^m$

where j is determined by finding the maximum j , such that $x_j < x$ and then

$$m = \frac{x - x_j}{T}$$

Example

The following example is based on the Sample Basic database. It forecasts sales data for May through December, based on the trend of the same sales data from January through April. The method used is linear regression with no seasonal adjustment.

```
Sales (@TREND (Jan:Apr, , , , , May:Dec, LR) );
```

This example produces the following report:

Actual	Sales	West
	Colas	
	=====	
Jan	2339	
Feb	2298	
Mar	2313	
Apr	2332	
May	2319	
Jun	2318.4	
Jul	2317.8	
Aug	2317.2	
Sep	2316.6	
Oct	2316	
Nov	2315.4	
Dec	2314.8	
Year	27817.2	

See Also

- [@LIST](#)

@TRUNCATE

Removes the fractional part of *expression*, returning the integer.

Syntax

```
@TRUNCATE (expression)
```

Parameter Description

expression Single member specification, function, variable name, or other numeric expression, which returns a numeric value.

Example

In the following example, Total Sales is calculated by (1) taking the sum of the values for Direct Sales and Other Sales and (2) truncating the summed values.

```
"Total Sales" = @TRUNCATE(@SUM("Direct Sales":"Other Sales"));
```

This example produces the following report:

	Colas	New York	Actual
	Jan	Feb	Mar
	===	===	===
Direct Sales	678.557	645.874	675.299
Other Sales	411.299	389.554	423.547
Total Sales	1089	1035	1098

See Also

- [@REMAINDER](#)
- [@ROUND](#)

@UDA

Returns members based on a common attribute, which you have defined as a user-defined attribute (UDA) on the Essbase Server.

Syntax

```
@UDA (dimName, uda)
```

Parameter Description

dimName Name of the dimension with which the *uda* is associated.

uda Name of the user-defined attribute as it appears in the database outline.

Notes

You must type the UDA string exactly as it appears in the database outline.

Example

In the Sample Basic database:

@UDA (Market, "New Mkt")

Returns a list of members with the UDA of New Mkt.

See Also

- [@ISUDA](#)

@VAR

Calculates the variance (difference) between two members. The variance calculation recognizes the difference between accounts that are tagged in the database outline as "Expense" or "No Expense" and calculates the variance accordingly.

Syntax

@VAR (*mbrName1*, *mbrName2*)

Parameter

Description

mbrName1 and *mbrName2* Members from the same dimension whose variance results are to be calculated. The variance is derived by subtracting *mbrName2* values from *mbrName1*, unless an account is tagged as "Expense", in which case *mbrName1* values are subtracted from *mbrName2*.

Example

The following example is based on the Sample Basic database. The variance between Actual and Budget is calculated as follows:

Variance = @VAR(Actual,Budget);

Sales is defined as "No Expense", whereas COGS is tagged as "Expense". This example produces the following report:

	Year	Product	Market
	Sales	COGS	
	=====	=====	
Actual	400855	179336	
Budget	373080	158940	
Variance	27775	(20396)	

See Also

- [@VARPER](#)
- [@VARIANCE](#)
- [@VARIANCEP](#)

@VARPER

Calculates the percent variance (difference) between two members. The variance calculation recognizes the difference between accounts that are tagged in the database outline as "Expense" or "No Expense" and calculates the variance accordingly.

Syntax

@VARPER (*mbrName1*, *mbrName2*)

Parameter	Description
-----------	-------------

<i>mbrName1</i> and <i>mbrName2</i>	Members from the same dimension whose variance results are to be calculated. The percent variance is derived by taking the percent variance of <i>mbrName2</i> values from <i>mbrName1</i> , unless an account is tagged as "Expense", in which case <i>mbrName1</i> values are taken as a percent variance of <i>mbrName2</i> .
-------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Example

The following example is based on the Sample Basic database. The percent variance between Actual and Budget is calculated as follows:

```
Variance = @VARPER(Actual,Budget);
```

In this example Sales is defined as "No Expense", whereas COGS is tagged as "Expense". This example produces the following report:

	Year	Product	Market
	Sales	COGS	
	=====	=====	
Actual	400855	179336	
Budget	373080	158940	
Variance %	7.4	(12.8)	

See Also

- [@VAR](#)
- [@VARIANCE](#)
- [@VARIANCEP](#)

@VARIANCE

Calculates the statistical variance of the specified data set (*expList*). The calculation is based upon a sample of a population. Variance is a measure of the dispersion of a set of data points around their mean (average) value.

Syntax

@VARIANCE (SKIPNONE | SKIPMISSING | SKIPZERO | SKIPBOTH, *expList*)

Parameter	Description
-----------	-------------

SKIPNONE	Includes all cells specified in <i>expList</i> , regardless of their content, during calculation of the variance.
----------	-------------------------------------------------------------------------------------------------------------------

SKIPMISSING	Excludes all #MISSING values from <i>expList</i> during calculation of the variance.
-------------	--------------------------------------------------------------------------------------

SKIPZERO	Excludes all zero (0) values from <i>expList</i> during calculation of the variance.
----------	--------------------------------------------------------------------------------------

SKIPBOTH	Excludes all zero (0) values and #MISSING values from <i>expList</i> during calculation of the variance.
----------	----------------------------------------------------------------------------------------------------------

<i>expList</i>	Comma-delimited list of member specifications, variable names, functions, or numeric expressions. <i>expList</i> provides a list of numeric values across which the variance is calculated.
----------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Notes

- @VARIANCE is different from @VAR, which calculates the variance (difference) between two members.
- @VARIANCE assumes that *expList* represents a sample of the population. If you want *expList* to represent the entire population, use @VARIANCEP.
- @VARIANCE is calculated with the "unbiased" or "n-1" method.
- @VARIANCE uses the following formula:

$$\frac{n \sum x^2 - (\sum x)^2}{n(n-1)}$$

Example

The following example is based on the Sample Basic database. Assume that the Measures dimension contains an additional member, Sales Var. This example uses the @RANGE function to generate *expList*, and calculates the variance of the sales values for a product family.

```
FIX (Product)
"Sales Var" = @VARIANCE(SKIPBOTH, @RANGE(Sales, @CHILDREN(Product)));
ENDFIX
```

This example produces the following report:

		Jan	New York
		Actual	Budget
		=====	=====
Sales	Colas	678	640
	Root Beer	551	530
	Cream Soda	663	510
	Fruit Soda	587	620
	Diet Drinks	#MI	#MI
	Product	2479	2300
Sales Var	Product	3687.58	4166.67

See Also

- @VARIANCEP

@VARIANCEP

The @VARIANCEP() function calculates the statistical variance of the specified data set (*expList*). The calculation is based upon the entire population. Variance is a measure of the dispersion of a set of data points around their mean (average) value.

Syntax

```
@VARIANCEP (SKIPNONE | SKIPMISSING | SKIPZERO | SKIPBOTH, expList)
```

Parameter	Description
SKIPNONE	Includes all cells specified in <i>expList</i> , regardless of their content, during calculation of the variance.
SKIPMISSING	Excludes all #MISSING values from <i>expList</i> during calculation of the variance.
SKIPZERO	Excludes all zero (0) values from <i>expList</i> during calculation of the variance.
SKIPBOTH	Excludes all zero (0) values and #MISSING values from <i>expList</i> during calculation of the variance.
expList	Comma-delimited list of member specifications, variable names, functions, or numeric expressions. <i>expList</i> provides a list of numeric values across which the variance is calculated.

Notes

- @VARIANCEP is different from @VARPER, which calculates the percent variance (difference) between two members.
- @VARIANCEP assumes that *expList* represents the entire population. If you want *expList* to represent a sample of the population, use @VARIANCE.
- @VARIANCEP is calculated using the "biased" or "n" method.
- @VARIANCEP uses the following formula:

$$\frac{n \sum x^2 - (\sum x)^2}{n^2}$$

Example

The following example is based on the Sample Basic database. Assume that the Measures dimension contains an additional member, Sales Var. This example uses the @RANGE function to generate *expList* and calculates the variance of the sales values for a product family.

```
FIX (Product)
"Sales Var" = @VARIANCEP(SKIPBOTH,@RANGE(Sales,@CHILDREN(Product)));
ENDFIX
```

This example produces the following report:

		Jan	New York
		Actual	Budget
		=====	=====
Sales	Colas	678	640
	Root Beer	551	530
	Cream Soda	663	510
	Fruit Soda	587	620
	Diet Drinks	#MI	#MI
	Product	2479	2300
Sales Var	Product	2765.69	3125

See Also

- @VARIANCE

@WITHATTR

Returns all base members that are associated with an attribute or varying attribute that satisfies the conditions you specify. You can use operators such as >, <, =, and IN to specify conditions that must be met. @WITHATTR can be used as a parameter of another function, where that parameter is a list of members.

Syntax

```
@WITHATTR (dimName, "operator", value)
```

Parameter Description

dimName Single attribute dimension name or varying attribute dimension name.

operator Operator specification, which must be enclosed in quotation marks ("").

value A value that, in combination with the operator, defines the condition that must be met. The *value* can be an attribute member specification, a constant, or a date-format function (that is, @TODATE).

Notes

- A varying attribute cannot be included in a FIX command if no perspective is specified in the calculation script.
- The @WITHATTR function is a superset of the @ATTRIBUTE function. The following two formulas return the same member set:

```
@ATTRIBUTE (Bottle)
@WITHATTR ("Pkg Type", "=", Bottle)
```

However, the following formula can be performed only with @WITHATTR (not with @ATTRIBUTE) because you specify a condition:

```
@WITHATTR (Ounces, ">", "16")
```

- If you specify a date attribute with the @WITHATTR function, you must use the @TODATE function in the *string* parameter to convert the date string to a number. For more information, see the topic for the @TODATE function.
- The following operators are supported:

Operator	Description
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
= =	Equal to
<> or !=	Not equal to
IN	In

- The IN operator returns the base members that are associated with a subcategory of attributes in the attribute dimension. For example, in the Sample Basic database, @WITHATTR(Population, "IN", Medium) returns the base members that are associated with all attributes under the Medium parent member in the Population dimension.
- When using Boolean attributes with @WITHATTR, use only the actual Boolean attribute member name, or use 1 (for True or Yes) or 0 (for False or No). You cannot use True/Yes and False/No interchangeably.
- An operator may work differently with different attribute types. For example:
 - **Text**—@WITHATTR(Flavors, "<", Orange) returns base members with attributes that precede Orange in the alphabet; for example, Apple, Cranberry, Mango, and Oat, but not Peach or Strawberry.
 - **Boolean**—@WITHATTR(Caffeinated, "<", True) returns all base members that have Caffeinated set to False (or No). It does not return base members that do *not* have Caffeinated set to True (or Yes) or do not have a Caffeinated attribute at all. The behavior is similar for a formula like @WITHATTR(Caffeinated, "<>", True), which returns only base members with Caffeinated set to False.
 - **Date**—@WITHATTR("Intro Date", "<", @TODATE("mm-dd-yyyy", "07-26-2002")) returns all base members with date attributes that are *before* July 26, 2002.

Example

The following table shows examples, based on the Sample Basic database, for each type of operator:

Operator	Example	Result
>	@WITHATTR(Population, ">", "18000000")	Returns New York, California, and Texas
>=	@WITHATTR(Population, ">=", 10000000) where 10,000,000 is not a numeric attribute member, but a constant	Returns New York, Florida, California, Texas, Illinois, and Ohio
<	@WITHATTR(Ounces, "<", "16")	Returns Cola, Diet Cola, Old Fashioned, Sasparilla, and Diet Cream
<=	@WITHATTR("Intro Date", "<=", @TODATE("mm-dd-yyyy", "04-01-2002"))	Returns Cola, Diet Cola, Caffeine Free Cola, and Old Fashioned
= =	@WITHATTR("Pkg Type", "= =", Can)	Returns Cola, Diet Cola, and Diet Cream
<> or !=	@WITHATTR(Caffeinated, "<>", True)	Returns Caffeine Free Cola, Sasparilla, Birch Beer, Grape, Orange Strawberry
IN	@WITHATTR("Population", "IN", Medium)	Returns Massachusetts, Florida, Illinois, and Ohio

The following two examples show @WITHATTR used in a calculation script, based on the Sample Basic database:

```
/* To increase by 10% the price of products that are greater than
or equal to 20 ounces */
```

```
FIX (@WITHATTR(Ounces, ">=", "20"))
Price = Price * 1.1;
ENDFIX
```

/* To increase by 10% the marketing budget for products brought to market after a certain date */

```
FIX (@WITHATTR("Intro Date", ">",
@TODATE("mm-dd-yyyy", "06-26-1996")));
Marketing = Marketing * 1.1;
ENDFIX
```

See Also

- [@ATTRIBUTE](#)
- [@ATTRIBUTEVAL](#)
- [@TODATE](#)
- [SET SCAPERSPECTIVE](#)

@XRANGE

Returns the range of members between (and inclusive of) two specified single or cross-dimensional members at the same level.

For example, when you work with the Time and Scenario dimensions, you can use @XRANGE to return a member set combination of Time and Scenario instead of creating a dimension that combines the two (which creates many more individual members than necessary).

@XRANGE is a member set function. Member set functions return a list of members. @XRANGE can appear anywhere in a formula where a range can normally appear.

Syntax

```
@XRANGE (mbrName1, mbrName2)
```

Parameter Description

mbrName1 Any valid member name, member combination, or function that returns a single member.

mbrName2 Any valid member name, member combination, or function that returns a single member. If *mbrName1* is a cross-dimensional member (such as Actual->Jan), then *mbrName2* must be also, and the dimension order must match the order used in *mbrName1*.

Notes

- @XRANGE can be used only in these functions: @AVGRANGE, @SUMRANGE, @MINRANGE, @MINSRANGE, @MAXRANGE, @MAXSRANGE, @STDDEVRRANGE, @MOVSUM, @MOVAVG, @MOVMIN, @MOVMAX, @MOVMEAN, @SPLINE.
- The two arguments to @XRANGE can be either both single members or both cross-dimensional members. For example, @XRANGE (Actual->Jan, Budget) is invalid because

a single member and a cross dimensional member are used together. Both @XRANGE(Actual->Jan, Budget->Feb) and @XRANGE(Jan, Mar) are valid.

- The dimension order of members must match for both arguments. For example, @XRANGE(Actual->Jun, Jul->Budget) is invalid because the two member components are in different orders. @XRANGE(Actual->Jun, Budget->Jul) is valid.
- Although the syntax is correct, a function such as @XRANGE (Dec, Mar) is meaningless because it results in an empty set.
- The member components of each argument must be from the same level. For example, @XRANGE(Actual->Jun, Budget->Qtr1) is invalid because Jun and Qtr1 are not from the same level.

Example

The following examples are based on the Sample Basic database.

Example 1

Here is a very simple example using simple members to return the range between Jan and Mar.

```
@XRANGE(Jan, Mar)
```

This example returns the following members:

```
Jan  
Feb  
Mar
```

Example 2

Here is a very simple example using cross dimensional members to return the range between Actual, Jan and Budget, Mar:

```
@XRANGE (Actual->Jan, Budget->Mar)
```

This example returns the following members:

```
Actual, Jan  
Actual, Feb  
Actual, Mar  
Actual, Apr  
Actual, May  
Actual, Jun  
Actual, Jul  
Actual, Aug  
Actual, Sep  
Actual, Oct  
Actual, Nov  
Actual, Dec  
Budget, Jan  
Budget, Feb  
Budget, Mar
```

Example 3

This example is not based on the Sample Basic database. It is based on database that contains a dimension called Year that contains members for each year, from 2001 to 2003.

The following formula computes the average sales for all months between Mar of 2000 and Jan of 2001.

```
SalesAvg= @MOVAVG(Sales, 3, @XRANGE("2001"->Mar, "2003"->Jan));
```

This example returns the following members:

	Colas	New York Sales	Actual SalesAvg
		=====	=====
2000			
	Mar	678	678
	Apr	645	645
	May	675	666
	Jun	712	677.3
	Jul	756	714.3
	Aug	890	786
	Sep	924	856.7
	Oct	914	909.3
	Nov	912	916.7
	Dec	723	849.7
2001			
	Jan	647	760.7

See Also

- [@AVGRANGE](#)
- [@SUMRANGE](#)
- [@MINRANGE](#)
- [@MINSRANGE](#)
- [@MAXRANGE](#)
- [@MAXSRANGE](#)
- [@STDEV RANGE](#)
- [@MOV SUM](#)
- [@MOVAVG](#)
- [@MOV MIN](#)
- [@MOV MAX](#)
- [@MOV MED](#)
- [@SPLINE](#)

@XREF

Enables a database calculation to incorporate values from another Essbase database.

The following terminology is used to describe the @XREF function:

- **Data target:** the database on which the current calculation is running (that is, the database on which the @XREF call originates).

- Data source: the database that is queried by the @XREF function. This database may be remote (that is, on a different machine than the data target).
- Point of view: the member combination currently being calculated on the data target (that is, the member combination that identifies the left hand side of a calculation).

The @XREF function retrieves values from a data source to be used in a calculation on a data target. @XREF does not impose member and dimension mapping restrictions, which means that the data source and data target outlines can be different.

As arguments, this function takes a location alias, an implied list of members that represents the current point of view, and an optional list of members to qualify the @XREF query on the data source. The second argument (the members making up the current point of view) is implied; that is, these members are not specified as an @XREF parameter. An @XREF query that omits the third argument indicates that a given data point in the data target will be set to the same data point in the data source.

Syntax

```
@XREF (locationAlias [, mbrList])
```

Parameter	Description
-----------	-------------

locationAlias	A location alias for the data source. A location alias is a descriptor that identifies the data source. The location alias must be set on the database on which the calculation script will be run. The location alias is set by the database administrator and specifies a server, application, database, username, and password for the data source.
---------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

mbrList	Optional. A comma-delimited list of member names that qualify the @XREF query. The members you specify for <i>mbrList</i> are sent to the data source in addition to the members in the current point of view in the data target. The data source then constructs a member combination, using in order of precedence:
---------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

- The members specified in *mbrList*
- The members in the current point of view
- The top member in any unspecified dimensions in the data source

The *mbrList* parameter (1) modifies the point of view on the data target or (2) defines a specific point of view on the data source. For example, the following formula modifies the point of view on the data target:

```
2003 (2003->Jan->Inventory = @XREF (sourceDB, Dec) ;)
```

If the cube on the data source (*sourceDB*) contains data only from 2002, this formula sets Inventory for Jan in 2003 to the Inventory value for Dec from 2002.

The following formula defines a specific point of view on the data target:

```
Jan = @XREF (sourceDB, January) ;
```

Assume that the data target contains the member Jan, while the data source (*sourceDB*) contains the member January. This formula simply maps the member in the data target (Jan) with its corresponding member in the data source (January), and pulls January from *sourceDB*.

See Notes for more information about the *mbrList* parameter.

Notes

- An error is returned if the members supplied in *mbrList* do not exist in the data source.

- The number of data cells queried on the data source must match the number of data cells expected on the data target.
- The member list cannot contain functions that return more than one member. For example, the following formula is *not* valid:

```
West = @XREF(SourceDb, @LEVMBRS(Market,0));
```

- The member list cannot contain ranges. For example, the following formula is *not* valid:

```
West = @XREF(SourceDb, Jan:Mar);
```

- *mbrList* can contain attribute members. For example, if the data source classifies products based on a color attribute, the following formula would calculate the sum of the sales of all red products and would assign the result to member RedThings:

```
RedThings = @XREF(SourceDb, Sales, Red);
```

- *mbrList* can contain attribute operators. For example, the following formula calculates RedThings as the average sales of all red products:

```
RedThings = @XREF(SourceDb, Sales, Red, Average);
```

For more information on attributes, see the *Oracle Essbase Database Administrator's Guide*.

- Using this function in a calculation script disables parallel calculation.
- @XREF can query all types of members. For example, members retrieved from a data source can be Dynamic Calc members as well as attribute members. Keep in mind that all performance considerations that apply to dynamic and attribute calculations also apply to @XREF queries that depend on dynamic and attribute members. For more information, see the *Oracle Essbase Database Administrator's Guide*.
- Over the course of an @XREF calculation, data in the source database may change. @XREF does not incorporate changes made after the beginning of the calculation.
- @XREF is a top-down formula. For more information on top-down formulas, see the *Oracle Essbase Database Administrator's Guide*.
- For a member that does not exist in either the data source or the data target, @XREF returns the value of the top dimension, not the value #M1.
- If you are using the @PARENT function within @XREF: the @XREF function requires the @NAME function to be used around @PARENT. For example:

```
COGS=@XREF(Sample, @NAME(@PARENT(Product)), Sales);
```

Example

For this example, consider the following two databases:

Main Database

```
Year
  Qtr1
  Qtr2
Measures
  Sales
  Units
Product
  100
  100-10
```

100-20
Market
 East
 West
Scenario
 Budget
 Forecast

Inflation Rates Database

Year
 Qtr1
 Qtr2
Assumptions
 Inflation
 Deflation = Inflation * .5 (Dynamic Calc)
Country
 US
 Canada
 Europe

The following formula is associated with the Main Database:

```
Units = Units * @XREF(InflatDB, Inflation, US);
```

Where *InflatDB* is the location alias for the Inflation Rates Database and *Inflation* is the member for which a data value is retrieved from *InflatDB*.

In this example, Essbase calculates the following member combinations:

Units->Qtr1->100-10->East->Budget = Units->Qtr1->100-10->East->Budget * **Inflation->Qtr1->US**

Units->Qtr2->100-10->East->Budget = Units->Qtr2->100-10->East->Budget * **Inflation->Qtr2->US** and so on.

See Also

- [@XWRITE](#)

@XWRITE

Enables a database calculation to write values to another Essbase database, or to the same database.

The following terminology is used to describe the @XWRITE function:

- **Data source:** the database on which the current calculation is running (that is, the database on which the @XWRITE call originates).
- **Data target:** the database that is updated by the @XWRITE function. This database may be remote (that is, on a different machine than the data source).
- **Point of view:** the member combination currently being calculated on the data source.

The @XWRITE function writes to data blocks, either in the same database or in a remote database, while calculating a block in the current database. @XWRITE does not impose member

and dimension mapping restrictions, which means that the data source and data target outlines can be different.

As arguments, this function takes a location alias, an implied list of members that represents the current point of view, and an optional list of members to qualify @XWRITE on the data target. The second argument (the members making up the current point of view) is implied; that is, these members are not specified as an @XWRITE parameter. An @XWRITE that omits the third argument indicates that a given data point in the data source will be set to the same data point in the data target.

Syntax

```
@XWRITE (expression, locationAlias [, mbrList])
```

Parameter	Description
-----------	-------------

expression	A single member specification, variable name, or other numeric expression corresponding to the value to be stored.
------------	--------------------------------------------------------------------------------------------------------------------

locationAlias	A location alias for the data target. The location alias must be set on the database on which the calculation script will be run. The location alias is set by the database administrator and specifies a server, application, database, username, and password for the data target.
---------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

The same location alias can be used by both @XREF and @XWRITE. For @XREF, it represents the data source, and for @XWRITE it represents the data target.

For @XWRITE only, a reserved keyword @LOOPBACK can be used to write to the same database.

mbrList	Optional. A comma-delimited list of member names that qualify the @XWRITE operation. The members you specify for <i>mbrList</i> , in addition to the members in the current point of view in the data source, determine what is written to the data target. The data target is written to using the following calculation logic (in order of precedence):
---------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

- The members specified in *mbrList*
- The members in the current point of view
- The top member in any unspecified dimensions in the data target

Therefore, the remote member list is calculated and written using members from current point of view, overridden with members from the *mbrList* specified to @XWRITE, and if some dimensions are still absent at the data target, the top most dimension of the data target is used.

See Notes for more information about the *mbrList* parameter.

Notes

- This function is applicable only to block storage databases.
- An error is returned if the members supplied in *mbrList* do not exist in the data target.
- The member list cannot contain functions that return more than one member. For example @LEVMBRS (Market, 0).
- The member list cannot contain ranges.
- The member list cannot contain attribute members or attribute operators.
- Using this function in a calculation script disables parallel calculation.
- @XWRITE is a top-down formula. For more information on top-down formulas, see the *Oracle Essbase Database Administrator's Guide*.

- If you are using the @PARENT function within @XWRITE, the @XWRITE function requires the @NAME function to be used around @PARENT.
- @XWRITE to dynamic calc cells is not recommended; the data is calculated in memory, but not written.
- @XWRITE can be used in calculation scripts as well as outline member formulas.

Example

The following Sample Basic formula writes the 100-30 values into 100-20 on the same database.

```
FIX (East, Actual, Budget, Sales)
"100-30" (
@XWRITE("100-30", @loopback, "100-20");
)
ENDFIX
```

The following Sample Basic formula writes the 100-30 values into 100-20 on a remote database, Sample2 Basic, using the location alias "sam2basic" defined from Sample Basic to Sample2 Basic.

```
FIX (East, Actual, Budget, Sales)
"100-30" (
@XWRITE("100-30", sam2basic, "100-20");
)
ENDFIX
```

See Also

- [@XREF](#)

Custom-Defined Calculation Functions

To get you started in creating custom-defined functions for the Essbase calculator, a set of example statistical functions is provided with this release. These examples are compiled and included in the `essbase.jar` file, located in the `ESSBASEPATH\java\` directory.

For information about creating custom-defined functions, see the MaxL DDL [Create Function](#) statement. For more information about custom-defined functions, see the *Oracle Essbase Database Administrator's Guide*.

- [“Java Code Examples” on page 234](#)
- [“MaxL Registration Scripts” on page 264](#)

Java Code Examples

The Java code for examples of custom-defined functions is provided in the file `statisti.jav`, copied below. For more information about the classes, methods, and constants in the `statisti.jav` file, see the Oracle Essbase Statistics Java Package.

The code contained in the `statisti.jav` file is implemented in the `ESSBASEPATH\java\essbase.jar` file. The examples in the `statisti.jav` file use constants which are defined

in the `essbase.jar` file. To use the constants defined in these examples, you must import the Calculator class constants defined in the `essbase.jar` file.

- [“register.xml Sample Code” on page 264](#)
- [“drop.xml Sample Code” on page 277](#)
- [“reglobal.xml Sample Code” on page 279](#)

Statisti.jav

```
package com.hyperion.essbase.calculator;

/**
 * This class provides a set of simple statistical routines. Some of them
 * are present native in Essbase as well and some are not.
 * Contains:
 * <ul>
 * <li>min, max</li>
 * <li>sum, weighted sum</li>
 * <li>product, weighted product</li>
 * <li>average, weighted average</li>
 * <li>geometric mean, weighted geometric mean</li>
 * <li>harmonic mean, weighted harmonic mean</li>
 * <li>variance (var and varp), weighted variance</li>
 * <li>standard deviation (stdev and stdevp), weighted standard deviation</li>
 * <li>covariance, weighted covariance</li>
 * <li>correlation, weighted correlation</li>
 * <li>skewness, weighted skewness</li>
 * <li>kurtosis, weighted kurtosis</li>
 * <li>rank, mode, median, percentile, quartile</li>
 * </ul>
 */
public final class Statistics implements CalculatorConstants {

    /**
     * Computes minimum value of given sequence. Missing values are ignored
     * @param data data array
     * @return minimum value in the array
     */
    public static double min (double [] data) {
        int i, n = data.length;

        if (n == 0)
            return MISSG;

        double min = data [0];
        boolean flag = (min == MISSG);

        for (i=1; i<n; i++) {
            double d = data [i];
            if (d != MISSG) {
                if (flag) {
                    min = d;
                    flag = false;
                }
                else if (d < min) {

```

```

        min = d;
    }
}

return min;
}

/**
 * Computes maximum value of given sequence. Missing values are ignored.
 * @param data data array
 * @return maximum value in the array
 */
public static double max (double [] data) {
    int i, n = data.length;

    if (n == 0)
        return MISSG;

    double max = data [0];
    boolean flag = (max == MISSG);

    for (i=1; i<n; i++) {
        double d = data [i];
        if (d != MISSG) {
            if (flag) {
                max = d;
                flag = false;
            }
            else if (d > max) {
                max = d;
            }
        }
    }
    return max;
}

/**
 * Computes sum of a given sequence. Missing values are ignored (treated as 0)
 * @param data data array
 * @return sum of the data
 */
public static double sum (double [] data) {
    int i, n = data.length;

    double sum = MISSG;
    for (i=0; i<n; i++) {
        double d = data [i];
        if (d != MISSG) {
            sum = Calculator.add (sum, d);
        }
    }
    return sum;
}

/**
 * Computes weighted sum of a given sequence.

```

```

* Missing values are ignored (treated as 0)
* @param data data array
* @param weights weights
* @return weighted sum of the data
*/
public static double sum (double [] data, double [] weights) {
    int i, n = data.length;

    double sum = MISSG;

    for (i=0; i<n; i++) {
        double d = data [i], w = weights [i];
        if (d != MISSG && w != MISSG) {
            sum = Calculator.add (sum, d * w);
        }
    }
    return sum;
}

/**
* Computes product of a given sequence. Missing values are ignored (treated as 0)
* @param data data array
* @return product of the data
*/
public static double product (double [] data) {
    int i, n = data.length;

    if (n == 0)
        return MISSG;

    double product = 1.;
    boolean flag = false;
    for (i=0; i<n; i++) {
        double d = data [i];
        if (d != MISSG) {
            flag = true;
            product = product * d;
        }
    }

    if (!flag)
        return MISSG;

    return product;
}

/**
* Computes weighted product of a given sequence.
* Missing values are ignored (treated as 0)
* @param data data array
* @param weights weights
* @return weighted product of the data
*/
public static double product (double [] data, double [] weights) {
    int i, n = data.length;

    if (n == 0)

```

```

        return MISSG;

double product = 1.;
boolean flag = false;

for (i=0; i<n; i++) {
    double d = data [i], w = weights [i];
    if (d != MISSG && w != MISSG) {
        d = Calculator.pow (d, w);
        if (d != MISSG) {
            flag = true;
            product = product * d;
        }
    }
}
if (!flag)
    return MISSG;

return product;
}

/**
 * Computes count of non-missing values in a given sequence.
 * @param data data array
 * @return count of the non-missing data
 */
public static int count (double [] data) {
    int i, n = data.length;

    int count = 0;

    for (i=0; i<n; i++) {
        double d = data [i];
        if (d != MISSG) {
            count ++;
        }
    }
    return count;
}

/**
 * Computes count of a given sequence (with prescribed skip directive).
 * @param skip skip instruction; possible values are
 * <ul>
 * <li>SKIPNONE - nothing skipped </li>
 * <li>SKIPZERO - zeros skipped</li>
 * <li>SKIPMISSG - missing values skipped</li>
 * <li>SKIPBOTH - skip both zeros and missing values</li>
 * </ul> (defined in CalculatorConstants interface)
 * @param data data array
 * @return count of the data
 */
public static int count (int skip, double [] data) {
    int i, n = data.length;
    if (skip == SKIPNONE)
        return n;

```

```

    if (skip == SKIPMISSG)
        return count (data);
    boolean bZero = false, bMissg = false;

    bZero = (skip == SKIPZERO) || (skip == SKIPBOTH);
    bMissg = (skip == SKIPBOTH);

    int count = 0;

    for (i=0; i<n; i++) {
        double d = data [i];
        if ((bMissg && d == MISSG) || (bZero && d == 0.))
            continue;
        count ++;
    }
    return count;
}

/**
 * Computes the average value of a given sequence. Missing values are ignored.
 * @param data data array
 * @return average of the data
 */
public static double avg (double [] data) {
    int i, n = data.length;

    double sum = MISSG;
    int count = 0;

    for (i=0; i<n; i++) {
        double d = data [i];
        if (d != MISSG) {
            sum = Calculator.add (sum, d);
            count ++;
        }
    }

    if (count == 0)
        return MISSG;

    return sum / count;
}

/**
 * Computes the average value of a given sequence (with prescribed skip directive).
 * @param skip skip instruction; possible values are
 * <ul>
 * <li>SKIPNONE - nothing skipped </li>
 * <li>SKIPZERO - zeros skipped</li>
 * <li>SKIPMISSG - missing values skipped</li>
 * <li>SKIPBOTH - skip both zeros and missing values</li>
 * </ul> (defined in CalculatorConstants interface)
 * @param data data array
 * @return average of the data
 */
public static double avg (int skip, double [] data) {
    int i, n = data.length;

```

```

boolean bZero = false, bMissg = false;

if (skip == SKIPMISSG)
    return avg (data);

bZero = (skip == SKIPZERO) || (skip == SKIPBOTH);
bMissg = (skip == SKIPBOTH);

double sum = MISSG;
int count = 0;

for (i=0; i<n; i++) {
    double d = data [i];
    if ((bMissg && d == MISSG) || (bZero && d == 0.))
        continue;
    sum = Calculator.add (sum, d);
    count ++;
}

if (count == 0)
    return MISSG;

return sum / count;
}

/**
 * Computes weighted average of a given sequence. Missing values are ignored
 * @param data data array
 * @param weights weights
 * @return weighted average of the data
 */
public static double avg (double [] data, double [] weights) {
    int i, n = data.length;

    double sum = MISSG;
    double weight = MISSG;

    for (i=0; i<n; i++) {
        double d = data [i], w = weights [i];
        if (d != MISSG && w != MISSG) {
            sum = Calculator.add (sum, d * w);
            weight = Calculator.add (weight, w);
        }
    }

    if (sum == MISSG || weight == MISSG || weight == 0.)
        return MISSG;

    return sum / weight;
}

/**
 * Computes weighted average value of a given sequence
 * (with prescribed skip directive).
 * @param skip skip instruction; possible values are
 * <ul>
 * <li>SKIPNONE - nothing skipped </li>

```



```

* <li>SKIPZERO - zeros skipped</li>
* <li>SKIPMISSG - missing values skipped</li>
* <li>SKIPBOTH - skip both zeros and missing values</li>
* </ul> (defined in CalculatorConstants interface)
* @param data data array
* @param weights weights
* @return weighted average of the data
*/
public static double avg (int skip, double [] data, double [] weights) {
    int i, n = data.length;
    boolean bZero = false, bMissg = false;

    if (skip == SKIPMISSG)
        return avg (data, weights);

    bZero = (skip == SKIPZERO) || (skip == SKIPBOTH);
    bMissg = (skip == SKIPBOTH);

    double sum = MISSG;
    double weight = MISSG;

    for (i=0; i<n; i++) {
        double d = data [i], w = weights [i];
        if ((bMissg && d == MISSG) || (bZero && d == 0.))
            continue;

        if (w != MISSG)
            sum = Calculator.add (sum, d * w);
        weight = Calculator.add (weight, w);
    }

    if (sum == MISSG || weight == MISSG || weight == 0.)
        return MISSG;

    return sum / weight;
}

/**
* Computes the geometric average value of a given sequence.
* Missing values are ignored.
* @param data data array
* @return average of the data
*/
public static double geomean (double [] data) {
    int i, n = data.length;

    if (n == 0)
        return MISSG;

    double product = 1.;
    int count = 0;

    for (i=0; i<n; i++) {
        double d = data [i];
        if (d != MISSG) {
            product = product * d;
            count ++;
        }
    }

```

```

    }
}

if (count == 0)
    return MISSG;

return Math.pow (product, 1. / (double) count);
}

/**
 * Computes weighted geometric average of a given sequence.
 * Missing values are ignored
 * @param data data array
 * @param weights weights
 * @return weighted average of the data
 */
public static double geomean (double [] data, double [] weights) {
    int i, n = data.length;

    double product = 1.;
    double weight = MISSG;

    for (i=0; i<n; i++) {
        double d = data [i], w = weights [i];
        if (d != MISSG && w != MISSG) {
            product = product * Math.pow (d, w);
            weight = Calculator.add (weight, w);
        }
    }

    if (weight == MISSG || weight == 0.)
        return MISSG;

    return Math.pow (product, 1. / weight);
}

/**
 * Computes harmonic mean of a given sequence.
 * Missing values are ignored.
 * @param data data array
 * @return harmonic mean of the data
 */
public static double harmean (double [] data) {
    int i, n = data.length;

    if (n == 0)
        return MISSG;

    double sum = MISSG;
    int count = 0;

    for (i=0; i<n; i++) {
        double d = data [i];
        if (d != MISSG) {
            if (d == 0.)
                return MISSG;
            sum = sum + 1. / d;
        }
    }
}

```

```

        count ++;
    }
}

if (count == 0 || sum == 0.)
    return MISSG;

return count / sum;
}

/**
 * Computes weighted harmonic mean of a given sequence.
 * Missing values are ignored
 * @param data data array
 * @param weights weights
 * @return weighted harmonic mean of the data
 */
public static double harmean (double [] data, double [] weights) {
    int i, n = data.length;

    double sum = MISSG;
    double weight = MISSG;

    for (i=0; i<n; i++) {
        double d = data [i], w = weights [i];
        if (d != MISSG && w != MISSG) {
            if (d == 0.)
                return MISSG;
            sum = Calculator.add (sum, w / d);
            weight = Calculator.add (weight, w);
        }
    }

    if (sum == MISSG || sum == 0. || weight == MISSG)
        return MISSG;

    return weight / sum;
}

/**
 * Computes variance of a given sequence. Missing values are ignored
 * @param data data array
 * @return variance of the data
 */
public static double var (double [] data) {
    int i, n = data.length;

    double d, sum = MISSG, avg = MISSG;
    int count = 0;

    for (i=0; i<n; i++) {
        d = data [i];
        if (d != MISSG) {
            sum = Calculator.add (sum, d);
            count ++;
        }
    }
}

```

```

    if (count < 2)
        return MISSG;

    avg = sum / count;
    sum = 0.;
    for (i=0; i<n; i++) {
        d = data [i];
        if (d != MISSG) {
            d = d - avg;
            d = d * d;
            sum = sum + d;
        }
    }

    return (sum / (count - 1));
}

/**
 * Computes standard deviation of a given sequence. Missing values are ignored
 * @param data data array
 * @return stdev of the data
 */
public static double stdev (double [] data) {
    return Calculator.sqrt (var (data));
}

/**
 * Computes variance of a given sequence (with prescribed skip directive).
 * @param skip skip instruction; possible values are
 * <ul>
 * <li>SKIPNONE - nothing skipped </li>
 * <li>SKIPZERO - zeros skipped</li>
 * <li>SKIPMISSG - missing values skipped</li>
 * <li>SKIPBOTH - skip both zeros and missing values</li>
 * </ul> (defined in CalculatorConstants interface)
 * @param data data array
 * @return variance of the data
 */
public static double var (int skip, double [] data) {
    int i, n = data.length;
    boolean bZero = false, bMissg = false;

    if (skip == SKIPMISSG)
        return var (data);

    bZero = (skip == SKIPZERO) || (skip == SKIPBOTH);
    bMissg = (skip == SKIPBOTH);

    double d, sum = MISSG, avg = MISSG;
    int count = 0;

    for (i=0; i<n; i++) {
        d = data [i];
        if ((bMissg && d == MISSG) || (bZero && d == 0.))
            continue;
        sum = Calculator.add (sum, d);
    }
}

```

```

        count ++;
    }

    if (count < 2)
        return MISSG;

    avg = sum / count;
    sum = 0.;
    for (i=0; i<n; i++) {
        d = data [i];
        if ((bMissg && d == MISSG) || (bZero && d == 0.))
            continue;
        if (d == MISSG)
            d = - avg;
        else
            d = d - avg;
        d = d * d;
        sum = sum + d;
    }

    return (sum / (count - 1));
}

/**
 * Computes standard deviation of a given sequence
 * (with prescribed skip directive).
 * @param skip skip instruction; possible values are
 * <ul>
 * <li>SKIPNONE - nothing skipped </li>
 * <li>SKIPZERO - zeros skipped</li>
 * <li>SKIPMISSG - missing values skipped</li>
 * <li>SKIPBOTH - skip both zeros and missing values</li>
 * </ul> (defined in CalculatorConstants interface)
 * @param data data array
 * @return standard deviation of the data
 */
public static double stdev (int skip, double [] data) {
    return Calculator.sqrt (var (skip, data));
}

/**
 * Computes weighted variance of a given sequence. Missing values are ignored
 * @param data data array
 * @param weights weights
 * @return weighted variance of the data
 */
public static double var (double [] data, double [] weights) {
    int i, n = data.length;

    double d, sum = MISSG, avg = MISSG;
    double w, weight = MISSG;

    for (i=0; i<n; i++) {
        d = data [i];
        w = weights [i];
        if (d != MISSG && w != MISSG) {
            sum = Calculator.add (sum, d * w);

```

```

        weight = Calculator.add (weight, w);
    }
}

if (sum == MISSG || weight == MISSG || weight == 0. || weight == 1.)
    return MISSG;

avg = sum / weight;
sum = 0.;
for (i=0; i<n; i++) {
    d = data [i];
    w = weights [i];
    if (d == MISSG || w == MISSG)
        continue;
    d = d - avg;
    d = d * d * w;
    sum = sum + d;
}
return (sum / (weight - 1.));
}

/**
 * Computes weighted standard deviation of a given sequence.
 * Missing values are ignored
 * @param data data array
 * @param weights weights
 * @return weighted standard deviation of the data
 * (without taking missing values into account)
 */
public static double stdev (double [] data, double [] weights) {
    return Calculator.sqrt (var (data, weights));
}

/**
 * Computes weighted variance of a given sequence
 * (with prescribed skip directive).
 * @param skip skip instruction; possible values are
 * <ul>
 * <li>SKIPNONE - nothing skipped </li>
 * <li>SKIPZERO - zeros skipped</li>
 * <li>SKIPMISSG - missing values skipped</li>
 * <li>SKIPBOTH - skip both zeros and missing values</li>
 * </ul> (defined in CalculatorConstants interface)
 * @param data data array
 * @param weights weights
 * @return weighted variance of the data
 */
public static double var (int skip, double [] data, double [] weights) {
    int i, n = data.length;
    boolean bZero = false, bMissg = false;

    if (skip == SKIPMISSG)
        return var (data, weights);

    bZero = (skip == SKIPZERO) || (skip == SKIPBOTH);
    bMissg = (skip == SKIPBOTH);

```

```

double sum = MISSG, avg = MISSG;
double weight = MISSG;

for (i=0; i<n; i++) {
    double d = data [i], w = weights [i];
    if ((bMissg && d == MISSG) || (bZero && d == 0.))
        continue;

    if (d != MISSG && w != MISSG)
        sum = Calculator.add (sum, d * w);
    weight = Calculator.add (weight, w);
}

if (sum == MISSG || weight == MISSG || weight == 0. || weight == 1.)
    return MISSG;

avg = sum / weight;
sum = 0.;
for (i=0; i<n; i++) {
    double d = data [i], w = weights [i];
    if ((bMissg && d == MISSG) || (bZero && d == 0.))
        continue;

    if (w != MISSG) {
        if (d == MISSG)
            d = -avg;
        else
            d = d - avg;
        d = d * d * w;
        sum = sum + d;
    }
}
return Math.sqrt (sum / (weight - 1));
}

/**
 * Computes weighted standard deviation of a given sequence
 * (with prescribed skip directive).
 * @param skip skip instruction; possible values are
 * <ul>
 * <li>SKIPNONE - nothing skipped </li>
 * <li>SKIPZERO - zeros skipped</li>
 * <li>SKIPMISSG - missing values skipped</li>
 * <li>SKIPBOTH - skip both zeros and missing values</li>
 * </ul> (defined in CalculatorConstants interface)
 * @param data data array
 * @param weights weights
 * @return weighted standard deviation of the data
 */
public static double stdev (int skip, double [] data, double [] weights) {
    return Calculator.sqrt (var (skip, data, weights));
}

/**
 * Computes variance of a given sequence. Missing values are ignored
 * @param data data array
 * @return variance of the data

```

```

*/
public static double varp (double [] data) {
    int i, n = data.length;

    double sum = MISSG, avg = MISSG;
    int count = 0;

    for (i=0; i<n; i++) {
        double d = data [i];
        if (d != MISSG) {
            sum = Calculator.add (sum, d);
            count ++;
        }
    }

    if (count == 0)
        return MISSG;

    avg = sum / count;
    sum = 0.;
    for (i=0; i<n; i++) {
        double d = data [i];
        if (d != MISSG) {
            d = d - avg;
            d = d * d;
            sum = sum + d;
        }
    }
    return (sum / count);
}

/**
 * Computes stdevp of a given sequence. Missing values are ignored
 * @param data data array
 * @return stdevp of the data
 */
public static double stdevp (double [] data) {
    return Calculator.sqrt (varp (data));
}

/**
 * Computes variancep of a given sequence
 * (with prescribed skip directive).
 * @param skip skip instruction; possible values are
 * <ul>
 * <li>SKIPNONE - nothing skipped </li>
 * <li>SKIPZERO - zeros skipped</li>
 * <li>SKIPMISSG - missing values skipped</li>
 * <li>SKIPBOTH - skip both zeros and missing values</li>
 * </ul> (defined in CalculatorConstants interface)
 * @param data data array
 * @return variancep of the data
 */
public static double varp (int skip, double [] data) {
    int i, n = data.length;
    boolean bZero = false, bMissg = false;

```



```

    if (skip == SKIPMISSG)
        return varp (data);

    bZero = (skip == SKIPZERO) || (skip == SKIPBOTH);
    bMissg = (skip == SKIPBOTH);

    double sum = MISSG, avg = MISSG;
    int count = 0;

    for (i=0; i<n; i++) {
        double d = data [i];
        if ((bMissg && d == MISSG) || (bZero && d == 0.))
            continue;
        sum = Calculator.add (sum, d);
        count ++;
    }

    if (count == 0)
        return MISSG;

    avg = sum / count;
    sum = 0.;
    for (i=0; i<n; i++) {
        double d = data [i];
        if ((bMissg && d == MISSG) || (bZero && d == 0.))
            continue;
        if (d == MISSG)
            d = - avg;
        else
            d = d - avg;
        d = d * d;
        sum = sum + d;
    }
    return (sum / count);
}

/**
 * Computes stdevp of a given sequence
 * (with prescribed skip directive).
 * @param skip skip instruction; possible values are
 * <ul>
 * <li>SKIPNONE - nothing skipped </li>
 * <li>SKIPZERO - zeros skipped</li>
 * <li>SKIPMISSG - missing values skipped</li>
 * <li>SKIPBOTH - skip both zeros and missing values</li>
 * </ul> (defined in CalculatorConstants interface)
 * @param data data array
 * @return stdevp of the data
 */
public static double stdevp (int skip, double [] data) {
    return Calculator.sqrt (varp (skip, data));
}

/**
 * Computes weighted varp of a given sequence. Missing values are ignored
 * @param data data array
 * @param weights weights

```

```

    * @return weighted varp of the data
    */
public static double varp (double [] data, double [] weights) {
    int i, n = data.length;

    double sum = MISSG, avg = MISSG;
    double weight = MISSG;

    for (i=0; i<n; i++) {
        double d = data [i], w = weights [i];
        if (d != MISSG && w != MISSG) {
            sum = Calculator.add (sum, d * w);
            weight = Calculator.add (weight, w);
        }
    }

    if (sum == MISSG || weight == MISSG || weight == 0.)
        return MISSG;

    avg = sum / weight;
    sum = 0.;
    for (i=0; i<n; i++) {
        double d = data [i], w = weights [i];
        if (d == MISSG || w == MISSG)
            continue;
        d = d - avg;
        d = d * d * w;
        sum = sum + d;
    }

    return (sum / weight);
}

/**
 * Computes weighted standard deviation of a given sequence.
 * Missing values are ignored
 * @param data data array
 * @param weights weights
 * @return weighted standard deviation of the data
 */
public static double stdevp (double [] data, double [] weights) {
    return Calculator.sqrt (varp (data, weights));
}

/**
 * Computes weighted varp of a given sequence
 * (with prescribed skip directive).
 * @param skip skip instruction; possible values are
 * <ul>
 * <li>SKIPNONE - nothing skipped </li>
 * <li>SKIPZERO - zeros skipped</li>
 * <li>SKIPMISSG - missing values skipped</li>
 * <li>SKIPBOTH - skip both zeros and missing values</li>
 * </ul> (defined in CalculatorConstants interface)
 * @param data data array
 * @param weights weights
 * @return weighted varp of the data

```

```

*/
public static double varp (int skip, double [] data, double [] weights) {
    int i, n = data.length;
    boolean bZero = false, bMissg = false;

    if (skip == SKIPMISSG)
        return varp (data, weights);

    bZero = (skip == SKIPZERO) || (skip == SKIPBOTH);
    bMissg = (skip == SKIPBOTH);

    double sum = MISSG, avg = MISSG;
    double weight = MISSG;

    for (i=0; i<n; i++) {
        double d = data [i], w = weights [i];
        if ((bMissg && d == MISSG) || (bZero && d == 0.))
            continue;

        if (d != MISSG && w != MISSG)
            sum = Calculator.add (sum, d * w);
        weight = Calculator.add (weight, w);
    }

    if (sum == MISSG || weight == MISSG || weight == 0.)
        return MISSG;

    avg = sum / weight;
    sum = 0.;
    for (i=0; i<n; i++) {
        double d = data [i], w = weights [i];
        if ((bMissg && d == MISSG) || (bZero && d == 0.))
            continue;

        if (w != MISSG) {
            if (d == MISSG)
                d = -avg;
            else
                d = d - avg;
            d = d * d * w;
            sum = sum + d;
        }
    }

    return (sum / weight);
}

/**
 * Computes weighted stdevp value of a given sequence
 * (with prescribed skip directive).
 * @param skip skip instruction; possible values are
 * <ul>
 * <li>SKIPNONE - nothing skipped </li>
 * <li>SKIPZERO - zeros skipped</li>
 * <li>SKIPMISSG - missing values skipped</li>
 * <li>SKIPBOTH - skip both zeros and missing values</li>
 * </ul> (defined in CalculatorConstants interface)

```

```

* @param data data array
* @param weights weights
* @return weighted stdevp of the data
*/
public static double stdevp (int skip, double [] data, double [] weights) {
    return Calculator.sqrt (varp (skip, data, weights));
}

/**
* Computes covariance between two sequences.
* If a missing value is encountered in either of the sequences,
* the corresponding position is skipped in both of them.
* @param x first array
* @param y second array
* @return covariance
*/
public static double covariance (double [] x, double [] y) {
    int i, n = x.length;

    if (n == 0)
        return MISSG;

    double d1, d2, avg1 = MISSG, avg2 = MISSG;
    int count = 0;

    for (i=0; i<n; i++) {
        d1 = x [i];
        d2 = y [i];
        if (d1 != MISSG && d2 != MISSG) {
            avg1 = Calculator.add (avg1, d1);
            avg2 = Calculator.add (avg2, d2);
            count ++;
        }
    }

    if (count < 1)
        return MISSG;

    avg1 = avg1 / count;
    avg2 = avg2 / count;

    double covar = 0.;
    for (i=0; i<n; i++) {
        d1 = x [i];
        d2 = y [i];
        if (d1 != MISSG && d2 != MISSG) {
            d1 = d1 - avg1;
            d2 = d2 - avg2;
            covar = covar + d1 * d2;
        }
    }
    return covar / count;
}

/**
* Computes weighted covariance between two sequences
* If a missing value is encountered in either of the sequences,

```

```

* the corresponding position is skipped in both of them.
* @param x first array
* @param y second array
* @return correlation
*/
public static double covariance (double [] x, double [] y, double [] weights) {
    int i, n = x.length;

    if (n == 0)
        return MISSG;

    double d1, d2, avg1 = MISSG, avg2 = MISSG;
    double w, weight = MISSG;

    for (i=0; i<n; i++) {
        d1 = x [i];
        d2 = y [i];
        w = weights [i];
        if (d1 != MISSG && d2 != MISSG && w != MISSG) {
            avg1 = Calculator.add (avg1, d1 * w);
            avg2 = Calculator.add (avg2, d2 * w);
            weight = Calculator.add (weight, w);
        }
    }

    if (avg1 == MISSG || weight == MISSG || weight == 0.)
        return MISSG;

    avg1 = avg1 / weight;
    avg2 = avg2 / weight;

    double covar = 0.;
    for (i=0; i<n; i++) {
        d1 = x [i];
        d2 = y [i];
        w = weights [i];
        if (d1 != MISSG && d2 != MISSG && w != MISSG) {
            d1 = d1 - avg1;
            d2 = d2 - avg2;
            covar = covar + w * d1 * d2;
        }
    }
    return covar / weight;
}

/**
* Computes correlation between two sequences
* If a missing value is encountered in either of the sequences,
* the corresponding position is skipped in both of them.
* @param x first array
* @param y second array
* @return correlation
*/
public static double correlation (double [] x, double [] y) {
    int i, n = x.length;

    if (n == 0)

```

```

        return MISSG;

double d1, d2, avg1 = MISSG, avg2 = MISSG;
int count = 0;

for (i=0; i<n; i++) {
    d1 = x [i];
    d2 = y [i];
    if (d1 != MISSG && d2 != MISSG) {
        avg1 = Calculator.add (avg1, d1);
        avg2 = Calculator.add (avg2, d2);
        count ++;
    }
}

if (count < 2)
    return MISSG;

avg1 = avg1 / count;
avg2 = avg2 / count;

double stdev1 = 0.;
double stdev2 = 0.;
double covar = 0.;
for (i=0; i<n; i++) {
    d1 = x [i];
    d2 = y [i];
    if (d1 != MISSG && d2 != MISSG) {
        d1 = d1 - avg1;
        d2 = d2 - avg2;
        covar = covar + d1 * d2;
        stdev1 = stdev1 + d1 * d1;
        stdev2 = stdev2 + d2 * d2;
    }
}

stdev1 = Math.sqrt (stdev1 / (count - 1));
stdev2 = Math.sqrt (stdev2 / (count - 1));
covar = covar / count;

return covar / (stdev1 * stdev2);
}

/**
 * Computes weighted correlation between two sequences
 * If a missing value is encountered in either of the sequences,
 * the corresponding position is skipped in both of them.
 * @param x first array
 * @param y second array
 * @return correlation
 */
public static double correlation (double [] x, double [] y, double [] weights) {
    int i, n = x.length;

    if (n == 0)
        return MISSG;

```

```

double d1, d2, avg1 = MISSG, avg2 = MISSG;
double w, weight = MISSG;

for (i=0; i<n; i++) {
    d1 = x [i];
    d2 = y [i];
    w = weights [i];
    if (d1 != MISSG && d2 != MISSG && w != MISSG) {
        avg1 = Calculator.add (avg1, d1 * w);
        avg2 = Calculator.add (avg2, d2 * w);
        weight = Calculator.add (weight, w);
    }
}

if (avg1 == MISSG || weight == MISSG || weight == 0. || weight == 1.)
    return MISSG;

avg1 = avg1 / weight;
avg2 = avg2 / weight;

double stdev1 = 0.;
double stdev2 = 0.;
double covar = 0.;
for (i=0; i<n; i++) {
    d1 = x [i];
    d2 = y [i];
    w = weights [i];
    if (d1 != MISSG && d2 != MISSG && w != MISSG) {
        d1 = d1 - avg1;
        d2 = d2 - avg2;
        covar = covar + w * d1 * d2;
        stdev1 = stdev1 + w * d1 * d1;
        stdev2 = stdev2 + w * d2 * d2;
    }
}

stdev1 = Math.sqrt (stdev1 / (weight - 1.));
stdev2 = Math.sqrt (stdev2 / (weight - 1.));
covar = covar / weight;

return covar / (stdev1 * stdev2);
}

/**
 * Computes skewness of a sequence. Missing values are skipped
 * @param data data array
 * @return scewness of the sequence
 */
public static double skew (double [] data) {
    int i, n = data.length;

    if (n == 0)
        return MISSG;

    double d, avg = MISSG;
    int count = 0;

```

```

for (i=0; i<n; i++) {
    d = data [i];
    if (d != MISSG) {
        avg = Calculator.add (avg, d);
        count ++;
    }
}

if (count < 3)
    return MISSG;

avg = avg / count;

double stdev = 0.;
for (i=0; i<n; i++) {
    d = data [i];
    if (d != MISSG) {
        d = d - avg;
        stdev = stdev + d * d;
    }
}

stdev = Math.sqrt (stdev / (count - 1));

if (stdev == 0.)
    return MISSG;
double skew = 0.;
for (i=0; i<n; i++) {
    d = data [i];
    if (d != MISSG) {
        d = d - avg;
        d = d / stdev;
        skew = skew + d * d * d;
    }
}

return skew * count / ((count - 1) * (count - 2));
}

/**
 * Computes weighted skewness of a sequence. Missing values are ignored
 * @param data data array
 * @return skewness of the sequence
 */
public static double skew (double [] data, double [] weights) {
    int i, n = data.length;

    if (n == 0)
        return MISSG;

    double d, avg = MISSG;
    double w, weight = MISSG;

    for (i=0; i<n; i++) {
        d = data [i];
        w = weights [i];
        if (d != MISSG && w != MISSG) {

```



```

        avg = Calculator.add (avg, w * d);
        weight = Calculator.add (weight, w);
    }
}

if (avg == MISSG || weight == MISSG || weight == 0. || weight == 1. || weight == 2.)
    return MISSG;

avg = avg / weight;

double stdev = 0.;
for (i=0; i<n; i++) {
    d = data [i];
    w = weights [i];
    if (d != MISSG && w != MISSG) {
        d = d - avg;
        stdev = stdev + w * d * d;
    }
}

stdev = Math.sqrt (stdev / (weight - 1));

if (stdev == 0.)
    return MISSG;
double skew = 0.;
for (i=0; i<n; i++) {
    d = data [i];
    w = weights [i];
    if (d != MISSG && w != MISSG) {
        d = d - avg;
        d = d / stdev;
        skew = skew + w * d * d * d;
    }
}

return skew * weight / ((weight - 1.) * (weight - 2.));
}

/**
 * Computes kurtosis of a sequence. Missing values are skipped
 * @param data data array
 * @return kurtosis of the sequence
 */
public static double kurt (double [] data) {
    int i, n = data.length;

    if (n == 0)
        return MISSG;

    double d, avg = MISSG;
    int count = 0;

    for (i=0; i<n; i++) {
        d = data [i];
        if (d != MISSG) {
            avg = Calculator.add (avg, d);
            count ++;

```

```

    }
}

if (count < 4)
    return MISSG;

avg = avg / count;

double stdev = 0.;
for (i=0; i<n; i++) {
    d = data [i];
    if (d != MISSG) {
        d = d - avg;
        stdev = stdev + d * d;
    }
}

stdev = Math.sqrt (stdev / (count - 1));

if (stdev == 0.)
    return MISSG;

double kurt = 0.;
for (i=0; i<n; i++) {
    d = data [i];
    if (d != MISSG) {
        d = d - avg;
        d = d / stdev;
        kurt = kurt + d * d * d * d;
    }
}

kurt = kurt * count * (count + 1) / (count - 1) - 3 * (count - 1) * (count - 1);
return kurt / ((count - 2) * (count - 3));
}

/**
 * Computes weighted kurtosis of a sequence. Missing values are ignored
 * @param x data array
 * @return kurtosis of the sequence
 */
public static double kurt (double [] data, double [] weights) {
    int i, n = data.length;

    if (n == 0)
        return MISSG;

    double d, avg = MISSG;
    double w, weight = MISSG;

    for (i=0; i<n; i++) {
        d = data [i];
        w = weights [i];
        if (d != MISSG && w != MISSG) {
            avg = Calculator.add (avg, w * d);
            weight = Calculator.add (weight, w);
        }
    }
}

```

```

    }

    if (avg == MISSG || weight == MISSG || weight == 0. ||
        weight == 1. || weight == 2. || weight == 3.)
        return MISSG;

    avg = avg / weight;

    double stdev = 0.;
    for (i=0; i<n; i++) {
        d = data [i];
        w = weights [i];
        if (d != MISSG && w != MISSG) {
            d = d - avg;
            stdev = stdev + w * d * d;
        }
    }

    stdev = Math.sqrt (stdev / (weight - 1));

    if (stdev == 0.)
        return MISSG;

    double kurt = 0.;
    for (i=0; i<n; i++) {
        d = data [i];
        w = weights [i];
        if (d != MISSG && w != MISSG) {
            d = d - avg;
            d = d / stdev;
            kurt = kurt + w * d * d * d * d;
        }
    }

    kurt = kurt * weight * (weight + 1.) / (weight - 1.) -
        3 * (weight - 1.) * (weight - 1.);
    return kurt / ((weight - 2.) * (weight - 3.));
}

/**
 * Computes rank of a value relative to a given sequence.
 * Missing elements in the sequence are ignored. Rank is 1-based.
 * Missing value is not ranked.
 * @param value value to be ranked
 * @param data array of data
 * @return rank in the sequence as a double
 */
public static double rank (double value, double [] data) {
    int i = 0, n = data.length;
    double d;
    int rank;

    if (value == MISSG)
        return MISSG;

    double [] ddd = new double [n];

```

```

int j = 0;
for (i=0; i<n; i++) {
    d = data [i];
    if (d != MISSG) {
        ddd [j] = d;
        j ++;
    }
}
n = j;
if (n == 0)
    return MISSG;

if (n == 1) {
    if (ddd [0] > value)
        return 2.;
    else
        return 1.;
}

Calculator.sort (ddd, 0, n-1);

rank = 1;
while (ddd [n - rank] > value) {
    rank++;
    if (rank > n)
        break;
}
return (double) rank;
}

/**
 * Computes mode of a sequence. Missing values are ignored
 * @param data array of data
 * @return mode of the sequence
 */
public static double mode (double [] data) {
    int i, j, n = data.length, maxFreq, freq;
    double d, mode;
    double [] ddd = new double [n];

    j = 0;
    for (i=0; i<n; i++) {
        if (data [i] != MISSG) {
            ddd [j] = data [i];
            j ++;
        }
    }
    n = j;
    if (n == 0)
        return MISSG;

    if (n == 1)
        return ddd [0];

    Calculator.sort (ddd, 0, n-1);

    mode = ddd [0];
}

```

```

maxFreq = 1;
while (i < n-1) {
    freq = 1;
    d = ddd [i];
    i++;
    while (ddd [i] == d) {
        freq++;
        i++;
        if (i >= n)
            break;
    }
    if (freq > maxFreq) {
        maxFreq = freq;
        mode = d;
    }
}
return mode;
}

/**
 * Computes median of a sequence. Missing values are ignored
 * @param data data array
 * @result median of the sequence
 */
public static double median (double [] data) {
    int i, j, n = data.length;
    int midIndex;
    double median;
    double [] ddd = new double [n];

    j = 0;
    for (i=0; i<n; i++) {
        if (data [i] != MISSG) {
            ddd [j] = data [i];
            j ++;
        }
    }
    n = j;

    if (n == 0)
        return MISSG;

    Calculator.sort (ddd, 0, n - 1);

    midIndex = n / 2;
    if (n % 2 == 0) {
        /* Average of the two middle numbers */
        median = (ddd [midIndex] + ddd [midIndex - 1]) / 2;
    }
    else {
        median = ddd [midIndex];
    }
    return median;
}

/**
 * Computes percentile of a sequence. Missing values are ignored

```

```

* @param percent percent value
* @param data double array
* @result percentile of the sequence
*/
public static double percentile (double percent, double [] data) {
    int i, j, n = data.length;
    int midIndex;
    double median, temp;
    double [] ddd = new double [n];

    j = 0;
    for (i=0; i<n; i++) {
        if (data [i] != MISSG) {
            ddd [j] = data [i];
            j ++;
        }
    }
    n = j;

    if (n == 0)
        return MISSG;

    Calculator.sort (ddd, 0, n-1);

    if (percent == 0.)
        return ddd [0];

    if (percent == 1.)
        return ddd [n-1];

    temp = percent * (double) n;
    median = Math.floor (temp);
    midIndex = (int) median;

    if (median != temp) {
        temp -= median;
        median = ddd [midIndex-1];
        median += (ddd [midIndex] - median) * temp;
    }
    else {
        median = ddd [midIndex];
    }
    return median;
}

/**
* Computes percentile of a part of a sequence. Missing values are ignored
* @param percent percent value
* @param size size to use
* @param data data array
* @result percentile of the subsequence
*/
public static double percentile (double percent, int size, double [] data) {
    int i, j, n = data.length;
    if (n > size)
        n = size;
    int midIndex;

```

```

double median, temp;
double [] ddd = new double [n];

j = 0;
for (i=0; i<n; i++) {
    if (data [i] != MISSG) {
        ddd [j] = data [i];
        j++;
    }
}

n = j;

if (n == 0)
    return MISSG;

Calculator.sort (ddd, 0, n-1);

if (percent == 0.)
    return ddd [0];

if (percent == 1.)
    return ddd [n-1];

temp = percent * (double) n;
median = Math.floor (temp);
midIndex = (int) median;

if (median != temp) {
    temp -= median;
    median = ddd [midIndex-1];
    median += (ddd [midIndex] - median) * temp;
}
else {
    median = ddd [midIndex];
}
return median;
}

/**
 * Computes quartile of a sequence. Missing values are ignored
 * @param quart indicates which value to return
 * Possible values are:
 * <ul>
 * <li>0 - return minimum</li>
 * <li>1 - return 25% percentile</li>
 * <li>2 - return median</li>
 * <li>3 - return 75% percentile</li>
 * <li>4 - return maximum</li>
 * </ul>
 * @param data double array
 * @result quartile of the sequence
 */
public static double quartile (int quart, double [] data) {
    switch (quart) {
        case 0:
            return min (data);

```

```

    case 1:
        return percentile (0.25, data);
    case 2:
        return median (data);
    case 3:
        return percentile (0.75, data);
    case 4:
        return max (data);
    default:
        return MISSG;
    }
}
}

```

MaxL Registration Scripts

Sample scripts for registering and dropping the example custom-defined functions are provided in the following files, located in the following directory of this documentation: `samples\cdf\examples`:

- `register.mxl`—To register the functions locally in an application (see [register.mxl Sample Code](#)).
- `drop.mxl`—To drop the functions (if they were registered locally) (see [drop.mxl Sample Code](#)).
- `reglobal.mxl` To register the functions globally (see [reglobal.mxl Sample Code](#)).

The sample files can be viewed or modified in any text editor. For more information about registering custom-defined functions, see the *Oracle Essbase Database Administrator's Guide*.

register.mxl Sample Code

```

/* <maxl version="11.1.1" encoding="UTF-8"/> */

/**
 * This script registers methods of the class Statistics as custom-defined functions
 * for a specified application
 * Usage: Log in to MaxL Shell, then call: msh register.mxl appname
 */

/**
 * Register function average
 */
CREATE MACRO $1.'@JAVG' (GROUP)
AS '@_JAVG(@@S)'
SPEC '@JAVG(expList)'
COMMENT 'Computes the average of non-missing values in a data set (expList)';

CREATE FUNCTION $1.'@_JAVG'
AS 'com.hyperion.essbase.calculator.Statistics.avg(double [])';

```



```

/**
 * Register function weighted average
 */
CREATE FUNCTION $1.'_JAVGW'
AS 'com.hyperion.essbase.calculator.Statistics.avg(double [],double [])'
SPEC '@JAVGW(@LIST(expList), @LIST(weightExpList))'
COMMENT 'Computes the weighted average of non-missing values in a data set (expList)';

/**
 * Register functions average and weighted average with a skip instruction.
 * These functions will be used through macros, so no spec/comment specified.
 * Since these functions will not be used directly, the names start with '@_'.
 */
CREATE FUNCTION $1.'_JAVGS'
AS 'com.hyperion.essbase.calculator.Statistics.avg(int,double [])';
CREATE FUNCTION $1.'_JAVGWS'
AS 'com.hyperion.essbase.calculator.Statistics.avg(int,double [],double [])';

/**
 * Register macro for average with a skip instruction
 */
CREATE MACRO $1.'_JAVGS' (SINGLE,GROUP)
AS
'@@IFSTRCMP (@@1, SKIPNONE)
  @_JAVGS (0, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPMISSING)
  @_JAVGS (1, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPZERO)
  @_JAVGS (2, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPBOTH)
  @_JAVGS (3, @@2)
@@ELSE
  @@ERROR (@@L1, @_INVALIDSKIP)
@@ENDIF
@@ENDIF
@@ENDIF
'
SPEC '@JAVGS(SKIPNONE|SKIPZERO|SKIPMISSING|SKIPBOTH, expList)'
COMMENT 'Computes the average value of a data set (expList) with skip instructions';

/**
 * Register macro for weighted average with a skip instruction
 */
CREATE MACRO $1.'_JAVGWS' (SINGLE,SINGLE,SINGLE)
AS
'@@IFSTRCMP (@@1, SKIPNONE)
  @_JAVGWS (0, @@2, @@3)
@@ELSE
  @@IFSTRCMP (@@1, SKIPMISSING)

```

```

    @_JAVGWS (1, @@2, @@3)
@@ELSE
    @@IFSTRCMP (@@1, SKIPZERO)
        @_JAVGWS (2, @@2, @@3)
    @@ELSE
        @@IFSTRCMP (@@1, SKIPBOTH)
            @_JAVGS (3, @@2, @@3)
        @@ELSE
            @@ERROR (@@L1, @_INVALIDSKIP)
        @@ENDIF
    @@ENDIF
@@ENDIF
@@ENDIF'
SPEC '@JAVGWS(SKIPNONE|SKIPZERO|SKIPMISSING|SKIPBOTH, @LIST(expList),
@LIST(weightExpList))'
COMMENT 'Computes the weighted average value of a data set (expList) with skip
instructions';

/**
 * Register function correlation
 */
CREATE FUNCTION $1.'@JCORR'
AS 'com.hyperion.essbase.calculator.Statistics.correlation(double [],double [])'
SPEC '@JCORR(@LIST(expList1), @LIST(expList2))'
COMMENT 'Computes the correlation coefficient between two data sets (expList1 and
expList2)';

/**
 * Register function weighted correlation
 */
CREATE FUNCTION $1.'@JCORRW'
AS 'com.hyperion.essbase.calculator.Statistics.correlation(double [],double [],double
[])'
SPEC '@JCORRW(@LIST(expList1), @LIST(expList2), @LIST(weightExpList))'
COMMENT 'Computes the weighted correlation coefficient between two data sets (expList1
and expList2)';

/**
 * Register function count
 */
CREATE MACRO $1.'@JCOUNT' (GROUP)
AS '@_JCOUNT(@@S)'
SPEC '@JCOUNT(expList)'
COMMENT 'Computes the count of non-missing elements in a data set (expList)';

CREATE FUNCTION $1.'@_JCOUNT'
AS 'com.hyperion.essbase.calculator.Statistics.count(double [])';

/**
 * Register function count with a skip instruction.
 * This function will be used through macros, so no spec/comment specified.
 * Since this function will not be used directly, the name starts with '@_'.
 */
CREATE FUNCTION $1.'@_JCOUNTS'

```

```

AS 'com.hyperion.essbase.calculator.Statistics.count(int,double [])';

/**
 * Register macro for count with a skip instruction
 */
CREATE MACRO $1.'@JCOUNTS'(SINGLE,GROUP)
AS
'@@IFSTRCMP (@@1, SKIPNONE)
  @_JCOUNTS (0, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPMISSING)
  @_JCOUNTS (1, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPZERO)
  @_JCOUNTS (2, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPBOTH)
  @_JCOUNTS (3, @@2)
@@ELSE
  @@ERROR (@@L1, @_INVALIDSKIP)
@@ENDIF
@@ENDIF
@@ENDIF'
SPEC '@JCOUNTS(SKIPNONE|SKIPZERO|SKIPMISSING|SKIPBOTH, expList)'
COMMENT 'Computes the number of elements of a data set (expList) with skip
instructions';

/**
 * Register function covariance
 */
CREATE FUNCTION $1.'@JCOVAR'
AS 'com.hyperion.essbase.calculator.Statistics.covariance(double [],double [])'
SPEC '@JCOVAR(@LIST(expList1), @LIST(expList2))'
COMMENT 'Computes the covariance between two data sets (expList1 and expList2)';

/**
 * Register function weighted covariance
 */
CREATE FUNCTION $1.'@JCOVARW'
AS 'com.hyperion.essbase.calculator.Statistics.covariance(double [],double [],double
[])'
SPEC '@JCOVARW(@LIST(expList1), @LIST(expList2), @LIST(weightExpList))'
COMMENT 'Computes the weighted covariance between two data sets (expList1 and
expList2)';

/**
 * Register function geometric mean
 */
CREATE MACRO $1.'@JGEOMEAN'(GROUP)
AS '@_JGEOMEAN(@@S)'
SPEC '@JGEOMEAN(expList)'
COMMENT 'Computes the geometric mean of a data set (expList)';

```

```

CREATE FUNCTION $1.'@_JGEOMEAN'
AS 'com.hyperion.essbase.calculator.Statistics.geomean(double [])';

/**
 * Register function weighted geometric mean
 */
CREATE FUNCTION $1.'@JGEOMEANW'
AS 'com.hyperion.essbase.calculator.Statistics.geomean(double [],double [])'
SPEC '@JGEOMEANW(@LIST(expList), @LIST(weightExpList))'
COMMENT 'Computes the weighted geometric mean of a data set (expList)';

/**
 * Register function harmonic mean
 */
CREATE MACRO $1.'@JHARMEAN'(GROUP)
AS '@_JHARMEAN(@@S)'
SPEC '@JHARMEAN(expList)'
COMMENT 'Computes the harmonic mean of a data set (expList)';

CREATE FUNCTION $1.'@_JHARMEAN'
AS 'com.hyperion.essbase.calculator.Statistics.harmean(double [])';

/**
 * Register function weighted harmonic mean
 */
CREATE FUNCTION $1.'@JHARMEANW'
AS 'com.hyperion.essbase.calculator.Statistics.harmean(double [],double [])'
SPEC '@JHARMEANW(@LIST(expList), @LIST(weightExpList))'
COMMENT 'Computes the weighted harmonic mean of a data set (expList)';

/**
 * Register function kurtosis
 */
CREATE MACRO $1.'@JKURT'(GROUP)
AS '@_JKURT(@@S)'
SPEC '@JKURT(expList)'
COMMENT 'Computes the kurtosis of a data set (expList)';

CREATE FUNCTION $1.'@_JKURT'
AS 'com.hyperion.essbase.calculator.Statistics.kurt(double [])';

/**
 * Register function weighted kurtosis
 */
CREATE FUNCTION $1.'@JKURTW'
AS 'com.hyperion.essbase.calculator.Statistics.kurt(double [],double [])'
SPEC '@JKURTW(@LIST(expList), @LIST(weightExpList))'
COMMENT 'Computes the weighted kurtosis of a data set (expList)';

/**
 * Register function max
 * There is only one function with this name, so no need to specify the signature
 */

```

```

CREATE MACRO $1.'@JMAX'(GROUP)
AS '@_JMAX(@@S)'
SPEC '@JMAX(expList)'
COMMENT 'Computes the maximum of a data set (expList)';

CREATE FUNCTION $1.'@_JMAX'
AS 'com.hyperion.essbase.calculator.Statistics.max';

/**
 * Register function median
 * There is only one function with this name, so no need to specify the signature
 */
CREATE MACRO $1.'@JMEDIAN'(GROUP)
AS '@_JMEDIAN(@@S)'
SPEC '@JMEDIAN(expList)'
COMMENT 'Computes the median of a data set (expList)';

CREATE FUNCTION $1.'@_JMEDIAN'
AS 'com.hyperion.essbase.calculator.Statistics.median';

/**
 * Register function min
 * There is only one function with this name, so no need to specify the signature
 */
CREATE MACRO $1.'@JMIN'(GROUP)
AS '@_JMIN(@@S)'
SPEC '@JMIN(expList)'
COMMENT 'Computes the minimum of a data set (expList)';

CREATE FUNCTION $1.'@_JMIN'
AS 'com.hyperion.essbase.calculator.Statistics.min';

/**
 * Register function mode
 * There is only one function with this name, so no need to specify the signature
 */
CREATE MACRO $1.'@JMODE'(GROUP)
AS '@_JMODE(@@S)'
SPEC '@JMODE(expList)'
COMMENT 'Computes the mode of a data set (expList)';

CREATE FUNCTION $1.'@_JMODE'
AS 'com.hyperion.essbase.calculator.Statistics.mode';

/**
 * Register function percentile
 */
CREATE MACRO $1.'@JPTILE'(SINGLE, GROUP)
AS '@_JPTILE(@@1, @@SH1)'
SPEC '@JPTILE(percent, expList)'
COMMENT 'Computes the specified (percent) percentile of a data set (expList)';

CREATE FUNCTION $1.'@_JPTILE'
AS 'com.hyperion.essbase.calculator.Statistics.percentile(double, double [])';

/**
 * Register function product

```

```

*/
CREATE MACRO $1.'@JPROD'(GROUP)
AS '@_JPROD(@@S)'
SPEC '@JPROD(expList)'
COMMENT 'Computes the product of non-missing values in a data set (expList)';

CREATE FUNCTION $1.'@_JPROD'
AS 'com.hyperion.essbase.calculator.Statistics.product(double [])';

/**
 * Register function weighted product
 */
CREATE FUNCTION $1.'@JPRODW'
AS 'com.hyperion.essbase.calculator.Statistics.product(double [],double [])'
SPEC '@JPRODW(@LIST(expList), @LIST(weightExpList))'
COMMENT 'Computes the weighted product of non-missing values in a data set (expList)';

/**
 * Register function quartile
 * There is only one function with this name, so no need to specify the signature
 */
CREATE MACRO $1.'@JQ TILE'(SINGLE, GROUP)
AS '@_JQ TILE(@@1, @@SH1)'
SPEC '@JQ TILE(quart,expList)'
COMMENT 'Computes the specified (quart) quartile of a data set (expList)';

CREATE FUNCTION $1.'@_JQ TILE'
AS 'com.hyperion.essbase.calculator.Statistics.quartile';

/**
 * Register function rank
 * There is only one function with this name, so no need to specify the signature
 */
CREATE MACRO $1.'@JRANK'(SINGLE, GROUP)
AS '@_JRANK(@@1, @@SH1)'
SPEC '@JRANK(value,expList)'
COMMENT 'Computes the rank of a value in a data set (expList)';

CREATE FUNCTION $1.'@_JRANK'
AS 'com.hyperion.essbase.calculator.Statistics.rank';

/**
 * Register function skewness
 */
CREATE MACRO $1.'@JSKEW'(GROUP)
AS '@_JSKEW(@@S)'
SPEC '@JSKEW(expList)'
COMMENT 'Computes the skewness of a data set (expList)';

CREATE FUNCTION $1.'@JSKEW'
AS 'com.hyperion.essbase.calculator.Statistics.skew(double [])';

/**
 * Register function weighted skewness

```

```

*/
CREATE FUNCTION $1.'@JSKEWW'
AS 'com.hyperion.essbase.calculator.Statistics.skew(double [],double [])'
SPEC '@JSKEWW(@LIST(expList), @LIST(weightExpList))'
COMMENT 'Computes the weighted skewness of a data set (expList)';

/**
 * Register function stdev
 */
CREATE FUNCTION $1.'@JSTDEV' (GROUP)
AS '@_JSTDEV(@@S)'
SPEC '@JSTDEV(expList)'
COMMENT 'Computes the standard deviation of non-missing values in a data set (expList)';

CREATE FUNCTION $1.'@_JSTDEV'
AS 'com.hyperion.essbase.calculator.Statistics.stdev(double [])';

/**
 * Register function weighted stdev
 */
CREATE FUNCTION $1.'@JSTDEVW'
AS 'com.hyperion.essbase.calculator.Statistics.stdev(double [],double [])'
SPEC '@JSTDEVW(@LIST(expList), @LIST(weightExpList))'
COMMENT 'Computes the weighted standard deviation of non-missing values in a data set
(expList)';

/**
 * Register functions stdev and weighted stdev with a skip instruction.
 * These functions will be used through macros, so no spec/comment specified.
 * Since these functions will not be used directly, the names start with '@_'.
 */
CREATE FUNCTION $1.'@_JSTDEVS'
AS 'com.hyperion.essbase.calculator.Statistics.stdev(int,double [])';
CREATE FUNCTION $1.'@_JSTDEVWS'
AS 'com.hyperion.essbase.calculator.Statistics.stdev(int,double [],double [])';

/**
 * Register macro for stdev with a skip instruction
 */
CREATE MACRO $1.'@JSTDEVS' (SINGLE, GROUP)
AS
'@@IFSTRCMP (@@1, SKIPNONE)
  @_JSTDEVS (0, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPMISSING)
    @_JSTDEVS (1, @@2)
  @@ELSE
    @@IFSTRCMP (@@1, SKIPZERO)
      @_JSTDEVS (2, @@2)
    @@ELSE
      @@IFSTRCMP (@@1, SKIPBOTH)
        @_JSTDEVS (3, @@2)
      @@ELSE

```

```

        @@ERROR (@@L1, @_INVALIDSKIP)
    @@ENDIF
    @@ENDIF
    @@ENDIF
    @@ENDIF'
SPEC '@JSTDEVS(SKIPNONE|SKIPZERO|SKIPMISSING|SKIPBOTH, expList)'
COMMENT 'Computes the standard deviation value of a data set (expList) with skip
instructions';

/**
 * Register macro for weighted standard deviation with a skip instruction
 */
CREATE MACRO $1.'@JSTDEVWS'(SINGLE,SINGLE,SINGLE)
AS
'@@IFSTRCMP (@@1, SKIPNONE)
    @_JSTDEVWS (0, @@2, @@3)
@@ELSE
    @@IFSTRCMP (@@1, SKIPMISSING)
        @_JSTDEVWS (1, @@2, @@3)
    @@ELSE
        @@IFSTRCMP (@@1, SKIPZERO)
            @_JSTDEVWS (2, @@2, @@3)
        @@ELSE
            @@IFSTRCMP (@@1, SKIPBOTH)
                @_JSTDEVWS (3, @@2, @@3)
            @@ELSE
                @@ERROR (@@L1, @_INVALIDSKIP)
            @@ENDIF
        @@ENDIF
    @@ENDIF
    @@ENDIF'
SPEC '@JSTDEVWS(SKIPNONE|SKIPZERO|SKIPMISSING|SKIPBOTH, expList, weightExpList)'
COMMENT 'Computes the weighted standard deviation value of a data set (expList) with
skip instructions';

/**
 * Register function stdevp
 */
CREATE MACRO $1.'@JSTDEVP'(GROUP)
AS '@_JSTDEVP(@@S)'
SPEC '@JSTDEVP(expList)'
COMMENT 'Computes the standard deviation(p) of non-missing values in a data set
(expList)';

CREATE FUNCTION $1.'@JSTDEVP'
AS 'com.hyperion.essbase.calculator.Statistics.stdevp(double [])';

/**
 * Register function weighted stdevp
 */
CREATE FUNCTION $1.'@JSTDEVPW'
AS 'com.hyperion.essbase.calculator.Statistics.stdevp(double [],double [])'
SPEC '@JSTDEVPW(@LIST(expList), @LIST(weightExpList))'
COMMENT 'Computes the weighted standard deviation(p) of non-missing values in a data set

```



```

(expList)';

/**
 * Register functions stdevp and weighted stdevp with a skip instruction.
 * These functions will be used through macros, so no spec/comment specified.
 * Since these functions will not be used directly, the names start with '@_'.
 */
CREATE FUNCTION $1.'@_JSTDEVPS'
AS 'com.hyperion.essbase.calculator.Statistics.stdevp(int,double [])';
CREATE FUNCTION $1.'@_JSTDEVPWS'
AS 'com.hyperion.essbase.calculator.Statistics.stdevp(int,double [],double [])';

/**
 * Register macro for stdevp with a skip instruction
 */
CREATE MACRO $1.'@JSTDEVPS'(SINGLE,GROUP)
AS
'@@IFSTRCMP (@@1, SKIPNONE)
  @_JSTDEVPS (0, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPMISSING)
  @_JSTDEVPS (1, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPZERO)
  @_JSTDEVPS (2, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPBOTH)
  @_JSTDEVPS (3, @@2)
@@ELSE
  @@ERROR (@@L1, @_INVALIDSKIP)
@@ENDIF
@@ENDIF
@@ENDIF'
SPEC '@JSTDEVPS(SKIPNONE|SKIPZERO|SKIPMISSING|SKIPBOTH, expList)'
COMMENT 'Computes the standard deviation(p) value of a data set (expList) with skip
instructions';

/**
 * Register macro for weighted stdevp with a skip instruction
 */
CREATE MACRO $1.'@JSTDEVPWS'(SINGLE,SINGLE,SINGLE)
AS
'@@IFSTRCMP (@@1, SKIPNONE)
  @_JSTDEVPWS (0, @@2, @@3)
@@ELSE
  @@IFSTRCMP (@@1, SKIPMISSING)
  @_JSTDEVPWS (1, @@2, @@3)
@@ELSE
  @@IFSTRCMP (@@1, SKIPZERO)
  @_JSTDEVPWS (2, @@2, @@3)
@@ELSE
  @@IFSTRCMP (@@1, SKIPBOTH)
  @_JSTDEVPS (3, @@2, @@3)

```

```

        @@ELSE
            @@ERROR (@@L1, @_INVALIDSKIP)
        @@ENDIF
    @@ENDIF
    @@ENDIF
    @@ENDIF'
SPEC '@JSTDEVPWS(SKIPNONE|SKIPZERO|SKIPMISSING|SKIPBOTH, expList, weightExpList)'
COMMENT 'Computes the weighted standard deviation(p) value of a data set (expList) with
skip instructions';

/**
 * Register function sum
 */
CREATE MACRO $1.'@JSUM'(GROUP)
AS '@_JSUM(@@S)'
SPEC '@JSUM(expList)'
COMMENT 'Computes the sum of a data set (expList)';

CREATE FUNCTION $1.'@_JSUM'
AS 'com.hyperion.essbase.calculator.Statistics.sum(double [])';

/**
 * Register function weighted SUM
 */
CREATE FUNCTION $1.'@JSUMW'
AS 'com.hyperion.essbase.calculator.Statistics.sum(double [],double [])'
SPEC '@JSUMW(@LIST(expList), @LIST(weightExpList))'
COMMENT 'Computes the weighted sum of a data set (expList)';

/**
 * Register function var
 */
CREATE MACRO $1.'@JVAR'(GROUP)
AS '@_JVAR(@@S)'
SPEC '@JVAR(expList)'
COMMENT 'Computes the variance of non-missing values in a data set (expList)';

CREATE FUNCTION $1.'@_JVAR'
AS 'com.hyperion.essbase.calculator.Statistics.var(double [])';

/**
 * Register function weighted var
 */
CREATE FUNCTION $1.'@JVARW'
AS 'com.hyperion.essbase.calculator.Statistics.var(double [],double [])'
SPEC '@JVARW(@LIST(expList), @LIST(weightExpList))'
COMMENT 'Computes the weighted variance of non-missing values in a data set (expList)';

/**
 * Register functions var and weighted var with a skip instruction.
 * These functions will be used through macros, so no spec/comment specified.
 * Since these functions will not be used directly, the names start with '@_'.
 */

```

```

CREATE FUNCTION $1.'@_JVARs'
AS 'com.hyperion.essbase.calculator.Statistics.var(int,double [])';
CREATE FUNCTION $1.'@_JVARWS'
AS 'com.hyperion.essbase.calculator.Statistics.var(int,double [],double [])';

/**
 * Register macro for var with a skip instruction
 */
CREATE MACRO $1.'@JVARs'(SINGLE,GROUP)
AS
'@@IFSTRCMP (@@1, SKIPNONE)
  @_JVARs (0, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPMISSING)
  @_JVARs (1, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPZERO)
  @_JVARs (2, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPBOTH)
  @_JVARs (3, @@2)
@@ELSE
  @@ERROR (@@L1, @_INVALIDSKIP)
@@ENDIF
@@ENDIF
@@ENDIF'
SPEC '@JVARs(SKIPNONE|SKIPZERO|SKIPMISSING|SKIPBOTH, expList)'
COMMENT 'Computes the variance value of a data set (expList) with skip instructions';

/**
 * Register macro for weighted variance with a skip instruction
 */
CREATE MACRO $1.'@JVARWS'(SINGLE,SINGLE,SINGLE)
AS
'@@IFSTRCMP (@@1, SKIPNONE)
  @_JVARWS (0, @@2, @@3)
@@ELSE
  @@IFSTRCMP (@@1, SKIPMISSING)
  @_JVARWS (1, @@2, @@3)
@@ELSE
  @@IFSTRCMP (@@1, SKIPZERO)
  @_JVARWS (2, @@2, @@3)
@@ELSE
  @@IFSTRCMP (@@1, SKIPBOTH)
  @_JVARWS (3, @@2, @@3)
@@ELSE
  @@ERROR (@@L1, @_INVALIDSKIP)
@@ENDIF
@@ENDIF
@@ENDIF'
SPEC '@JVARWS(SKIPNONE|SKIPZERO|SKIPMISSING|SKIPBOTH, expList, weightExpList)'
COMMENT 'Computes the weighted variance value of a data set (expList) with skip
instructions';

```

```

/**
 * Register function varp
 */
CREATE MACRO $1.'@JVARP'(GROUP)
AS '@_JVARP(@@S)'
SPEC '@JVARP(expList)'
COMMENT 'Computes the variance(p) of non-missing values in a data set (expList)';

CREATE FUNCTION $1.'@_JVARP'
AS 'com.hyperion.essbase.calculator.Statistics.varp(double [])';

/**
 * Register function weighted varp
 */
CREATE FUNCTION $1.'@JVARPW'
AS 'com.hyperion.essbase.calculator.Statistics.varp(double [],double [])'
SPEC '@JVARPW(@LIST(expList), @LIST(weightExpList))'
COMMENT 'Computes the weighted variance(p) of non-missing values in a data set
(expList)';

/**
 * Register functions varp and weighted varp with a skip instruction.
 * These functions will be used through macros, so no spec/comment specified.
 * Since these functions will not be used directly, the names start with '@_'.
 */
CREATE FUNCTION $1.'@_JVARPS'
AS 'com.hyperion.essbase.calculator.Statistics.varp(int,double [])';
CREATE FUNCTION $1.'@_JVARPWS'
AS 'com.hyperion.essbase.calculator.Statistics.varp(int,double [],double [])';

/**
 * Register macro for varp with a skip instruction
 */
CREATE MACRO $1.'@JVARPS'(SINGLE,GROUP)
AS
'@@IFSTRCMP (@@1, SKIPNONE)
  @_JVARPS (0, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPMISSING)
  @_JVARPS (1, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPZERO)
  @_JVARPS (2, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPBOTH)
  @_JVARPS (3, @@2)
@@ELSE
  @@ERROR (@@L1, @_INVALIDSKIP)
@@ENDIF
@@ENDIF
@@ENDIF'
SPEC '@JVARPS(SKIPNONE|SKIPZERO|SKIPMISSING|SKIPBOTH, expList)'

```

```

COMMENT 'Computes the variance(p) value of a data set (expList) with skip instructions';

/**
 * Register macro for weighted varp with a skip instruction
 */
CREATE MACRO $1.'@JVARPWS' (SINGLE,SINGLE,SINGLE)
AS
'@@IFSTRCMP (@@1, SKIPNONE)
  @_JVARPWS (0, @@2, @@3)
@@ELSE
  @@IFSTRCMP (@@1, SKIPMISSING)
  @_JVARPWS (1, @@2, @@3)
@@ELSE
  @@IFSTRCMP (@@1, SKIPZERO)
  @_JVARPWS (2, @@2, @@3)
@@ELSE
  @@IFSTRCMP (@@1, SKIPBOTH)
  @_JVARPS (3, @@2, @@3)
@@ELSE
  @@ERROR (@@L1, @_INVALIDSKIP)
@@ENDIF
@@ENDIF
@@ENDIF'
SPEC '@JVARPWS(SKIPNONE|SKIPZERO|SKIPMISSING|SKIPBOTH, expList, weightExpList)'
COMMENT 'Computes the weighted variance(p) value of a data set (expList) with skip
instructions';

```

drop.mxl Sample Code

```

/* <maxl version="7.0.0" encoding="UTF-8"/> */

/**
 * This script deregisters methods of the class Statistics as custom-defined functions
 * for a specified application
 * Usage: Log in to MaxL Shell, then call: msh drop.mxl appname
 */

/**
 * Deregister all functions
 */
DROP FUNCTION $1.'@JAVG';
DROP FUNCTION $1.'@JAVGW';
DROP FUNCTION $1.'@_JAVGS';
DROP FUNCTION $1.'@_JAVGWS';
DROP MACRO $1.'@JAVGS';
DROP MACRO $1.'@JAVGWS';

DROP FUNCTION $1.'@JCORR';
DROP FUNCTION $1.'@JCORRW';

DROP FUNCTION $1.'@JCOUNT';
DROP FUNCTION $1.'@_JCOUNTS';
DROP MACRO $1.'@JCOUNTS';

```

```

DROP FUNCTION $1.'@JCOVAR';
DROP FUNCTION $1.'@JCOVARW';

DROP FUNCTION $1.'@JGEOMEAN';
DROP FUNCTION $1.'@JGEOMEANW';

DROP FUNCTION $1.'@JHARMEAN';
DROP FUNCTION $1.'@JHARMEANW';

DROP FUNCTION $1.'@JKURT';
DROP FUNCTION $1.'@JKURTW';

DROP FUNCTION $1.'@JMAX';

DROP FUNCTION $1.'@JMEDIAN';

DROP FUNCTION $1.'@JMIN';

DROP FUNCTION $1.'@JMODE';

DROP FUNCTION $1.'@JPTILE';

DROP FUNCTION $1.'@JPROD';
DROP FUNCTION $1.'@JPRODW';

DROP FUNCTION $1.'@JQFILE';

DROP FUNCTION $1.'@JRANK';

DROP FUNCTION $1.'@JSKEW';
DROP FUNCTION $1.'@JSKEWW';

DROP FUNCTION $1.'@JSTDEV';
DROP FUNCTION $1.'@JSTDEVW';
DROP FUNCTION $1.'@_JSTDEVS';
DROP FUNCTION $1.'@_JSTDEVWS';
DROP MACRO $1.'@JSTDEVS';
DROP MACRO $1.'@JSTDEVWS';

DROP FUNCTION $1.'@JSTDEVP';
DROP FUNCTION $1.'@JSTDEVPW';
DROP FUNCTION $1.'@_JSTDEVPS';
DROP FUNCTION $1.'@_JSTDEVPWS';
DROP MACRO $1.'@JSTDEVPS';
DROP MACRO $1.'@JSTDEVPWS';

DROP FUNCTION $1.'@JSUM';
DROP FUNCTION $1.'@JSUMW';

DROP FUNCTION $1.'@JVAR';
DROP FUNCTION $1.'@JVARW';
DROP FUNCTION $1.'@_JVAR';
DROP FUNCTION $1.'@_JVARWS';
DROP MACRO $1.'@JVAR';
DROP MACRO $1.'@JVARWS';

```

```

DROP FUNCTION $1.'@JVARP';
DROP FUNCTION $1.'@JVARPW';
DROP FUNCTION $1.'@_JVARPS';
DROP FUNCTION $1.'@_JVARPWS';
DROP MACRO $1.'@JVARPS';
DROP MACRO $1.'@JVARPWS';

/**
 * Restart the application
 */
ALTER SYSTEM UNLOAD APPLICATION $1;
ALTER SYSTEM LOAD APPLICATION $1;

```

reglobal.mxl Sample Code

```

/* <maxl version="11.1.1" encoding="UTF-8"/> */

/**
 * This script registers methods of the class Statistics as global custom-defined
 functions
 * Usage: Log in to MaxL Shell, then call: msh reglobal.mxl
 *
 */

/**
 * Register function average
 */
CREATE MACRO '@JAVG'(GROUP)
AS '@_JAVG(@@S)'
SPEC '@JAVG(expList)'
COMMENT 'Computes the average of non-missing values in a data set (expList)';

CREATE FUNCTION '@_JAVG'
AS 'com.hyperion.essbase.calculator.Statistics.avg(double [])';

/**
 * Register function weighted average
 */
CREATE FUNCTION '@JAVGW'
AS 'com.hyperion.essbase.calculator.Statistics.avg(double [],double [])'
SPEC '@JAVGW(@LIST(expList), @LIST(weightExpList))'
COMMENT 'Computes the weighted average of non-missing values in a data set (expList)';

/**
 * Register functions average and weighted average with a skip instruction.
 * These functions will be used through macros, so no spec/comment specified.
 * Since these functions will not be used directly, the names start with '@_'.
 */
CREATE FUNCTION '@_JAVGS'
AS 'com.hyperion.essbase.calculator.Statistics.avg(int,double [])';
CREATE FUNCTION '@_JAVGWS'
AS 'com.hyperion.essbase.calculator.Statistics.avg(int,double [],double [])';

```

```

/**
 * Register macro for average with a skip instruction
 */
CREATE MACRO '@JAVGS' (SINGLE, GROUP)
AS
'@@IFSTRCMP (@@1, SKIPNONE)
  @_JAVGS (0, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPMISSING)
  @_JAVGS (1, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPZERO)
  @_JAVGS (2, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPBOTH)
  @_JAVGS (3, @@2)
@@ELSE
  @@ERROR (@@L1, @_INVALIDSKIP)
@@ENDIF
@@ENDIF
@@ENDIF
@@ENDIF'
SPEC '@JAVGS(SKIPNONE|SKIPZERO|SKIPMISSING|SKIPBOTH, expList)'
COMMENT 'Computes the average value of a data set (expList) with skip instructions';

```

```

/**
 * Register macro for weighted average with a skip instruction
 */
CREATE MACRO '@JAVGWS' (SINGLE, SINGLE, SINGLE)
AS
'@@IFSTRCMP (@@1, SKIPNONE)
  @_JAVGWS (0, @@2, @@3)
@@ELSE
  @@IFSTRCMP (@@1, SKIPMISSING)
  @_JAVGWS (1, @@2, @@3)
@@ELSE
  @@IFSTRCMP (@@1, SKIPZERO)
  @_JAVGWS (2, @@2, @@3)
@@ELSE
  @@IFSTRCMP (@@1, SKIPBOTH)
  @_JAVGWS (3, @@2, @@3)
@@ELSE
  @@ERROR (@@L1, @_INVALIDSKIP)
@@ENDIF
@@ENDIF
@@ENDIF
@@ENDIF'
SPEC '@JAVGWS(SKIPNONE|SKIPZERO|SKIPMISSING|SKIPBOTH, @LIST(expList),
@LIST(weightExpList))'
COMMENT 'Computes the weighted average value of a data set (expList) with skip
instructions';

```

```

/**
 * Register function correlation
 */

```



```

CREATE FUNCTION '@JCORR'
AS 'com.hyperion.essbase.calculator.Statistics.correlation(double [],double [])'
SPEC '@JCORR(@LIST(expList1), @LIST(expList2))'
COMMENT 'Computes the correlation coefficient between two data sets (expList1 and
expList2)';

/**
 * Register function weighted correlation
 */
CREATE FUNCTION '@JCORRW'
AS 'com.hyperion.essbase.calculator.Statistics.correlation(double [],double [],double
[])'
SPEC '@JCORRW(@LIST(expList1), @LIST(expList2), @LIST(weightExpList))'
COMMENT 'Computes the weighted correlation coefficient between two data sets (expList1
and expList2)';

/**
 * Register function count
 */
CREATE MACRO '@JCOUNT'(GROUP)
AS '@_JCOUNT(@@S)'
SPEC '@JCOUNT(expList)'
COMMENT 'Computes the count of non-missing elements in a data set (expList)';

CREATE FUNCTION '@_JCOUNT'
AS 'com.hyperion.essbase.calculator.Statistics.count(double [])';

/**
 * Register function count with a skip instruction.
 * This function will be used through macros, so no spec/comment specified.
 * Since this function will not be used directly, the name starts with '@_'.
 */
CREATE FUNCTION '@_JCOUNTS'
AS 'com.hyperion.essbase.calculator.Statistics.count(int,double [])';

/**
 * Register macro for count with a skip instruction
 */
CREATE MACRO '@JCOUNTS'(SINGLE,GROUP)
AS
'@@IFSTRCMP (@@1, SKIPNONE)
  @_JCOUNTS (0, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPMISSING)
  @_JCOUNTS (1, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPZERO)
  @_JCOUNTS (2, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPBOTH)
  @_JCOUNTS (3, @@2)
@@ELSE
  @@ERROR (@@L1, @_INVALIDSKIP)
@@ENDIF

```

```

        @@ENDIF
    @@ENDIF
    @@ENDIF'
SPEC '@JCOUNTS(SKIPNONE|SKIPZERO|SKIPMISSING|SKIPBOTH, expList)'
COMMENT 'Computes the number of elements of a data set (expList) with skip
instructions';

/**
 * Register function covariance
 */
CREATE FUNCTION '@JCOVAR'
AS 'com.hyperion.essbase.calculator.Statistics.covariance(double [],double [])'
SPEC '@JCOVAR(@LIST(expList1), @LIST(expList2))'
COMMENT 'Computes the covariance between two data sets (expList1 and expList2)';

/**
 * Register function weighted covariance
 */
CREATE FUNCTION '@JCOVARW'
AS 'com.hyperion.essbase.calculator.Statistics.covariance(double [],double [],double
[])'
SPEC '@JCOVARW(@LIST(expList1), @LIST(expList2), @LIST(weightExpList))'
COMMENT 'Computes the weighted covariance between two data sets (expList1 and
expList2)';

/**
 * Register function geometric mean
 */
CREATE MACRO '@JGEOMEAN'(GROUP)
AS '@_JGEOMEAN(@@S)'
SPEC '@JGEOMEAN(expList)'
COMMENT 'Computes the geometric mean of a data set (expList)';

CREATE FUNCTION '@_JGEOMEAN'
AS 'com.hyperion.essbase.calculator.Statistics.geomean(double [])';

/**
 * Register function weighted geometric mean
 */
CREATE FUNCTION '@JGEOMEANW'
AS 'com.hyperion.essbase.calculator.Statistics.geomean(double [],double [])'
SPEC '@JGEOMEANW(@LIST(expList), @LIST(weightExpList))'
COMMENT 'Computes the weighted geometric mean of a data set (expList)';

/**
 * Register function harmonic mean
 */
CREATE MACRO '@JHARMEAN'(GROUP)
AS '@_JHARMEAN(@@S)'
SPEC '@JHARMEAN(expList)'
COMMENT 'Computes the harmonic mean of a data set (expList)';

CREATE FUNCTION '@_JHARMEAN'

```

```

AS 'com.hyperion.essbase.calculator.Statistics.harmean(double [])';

/**
 * Register function weighted harmonic mean
 */
CREATE FUNCTION '@JHARMEANW'
AS 'com.hyperion.essbase.calculator.Statistics.harmean(double [],double [])'
SPEC '@JHARMEANW(@LIST(expList), @LIST(weightExpList))'
COMMENT 'Computes the weighted harmonic mean of a data set (expList)';

/**
 * Register function kurtosis
 */
CREATE MACRO '@JKURT'(GROUP)
AS '@_JKURT(@@S)'
SPEC '@JKURT(expList)'
COMMENT 'Computes the kurtosis of a data set (expList)';

CREATE FUNCTION '@_JKURT'
AS 'com.hyperion.essbase.calculator.Statistics.kurt(double [])';

/**
 * Register function weighted kurtosis
 */
CREATE FUNCTION '@JKURTW'
AS 'com.hyperion.essbase.calculator.Statistics.kurt(double [],double [])'
SPEC '@JKURTW(@LIST(expList), @LIST(weightExpList))'
COMMENT 'Computes the weighted kurtosis of a data set (expList)';

/**
 * Register function max
 * There is only one function with this name, so no need to specify the signature
 */
CREATE MACRO '@JMAX'(GROUP)
AS '@_JMAX(@@S)'
SPEC '@JMAX(expList)'
COMMENT 'Computes the maximum of a data set (expList)';

CREATE FUNCTION '@_JMAX'
AS 'com.hyperion.essbase.calculator.Statistics.max';

/**
 * Register function median
 * There is only one function with this name, so no need to specify the signature
 */
CREATE MACRO '@JMEDIAN'(GROUP)
AS '@_JMEDIAN(@@S)'
SPEC '@JMEDIAN(expList)'
COMMENT 'Computes the median of a data set (expList)';

CREATE FUNCTION '@_JMEDIAN'
AS 'com.hyperion.essbase.calculator.Statistics.median';

/**
 * Register function min

```

```

* There is only one function with this name, so no need to specify the signature
*/
CREATE MACRO '@JMIN'(GROUP)
AS '@_JMIN(@@S)'
SPEC '@JMIN(expList)'
COMMENT 'Computes the minimum of a data set (expList)';

CREATE FUNCTION '@_JMIN'
AS 'com.hyperion.essbase.calculator.Statistics.min';

/**
* Register function mode
* There is only one function with this name, so no need to specify the signature
*/
CREATE MACRO '@JMODE'(GROUP)
AS '@_JMODE(@@S)'
SPEC '@JMODE(expList)'
COMMENT 'Computes the mode of a data set (expList)';

CREATE FUNCTION '@_JMODE'
AS 'com.hyperion.essbase.calculator.Statistics.mode';

/**
* Register function percentile
*/
CREATE MACRO '@JPTILE'(SINGLE, GROUP)
AS '@_JPTILE(@@1, @@SH1)'
SPEC '@JPTILE(percent,expList)'
COMMENT 'Computes the specified (percent) percentile of a data set (expList)';

CREATE FUNCTION '@_JPTILE'
AS 'com.hyperion.essbase.calculator.Statistics.percentile(double,double [])';

/**
* Register function product
*/
CREATE MACRO '@JPROD'(GROUP)
AS '@_JPROD(@@S)'
SPEC '@JPROD(expList)'
COMMENT 'Computes the product of non-missing values in a data set (expList)';

CREATE FUNCTION '@_JPROD'
AS 'com.hyperion.essbase.calculator.Statistics.product(double [])';

/**
* Register function weighted product
*/
CREATE FUNCTION '@JPRODW'
AS 'com.hyperion.essbase.calculator.Statistics.product(double [],double [])'
SPEC '@JPRODW(@LIST(expList), @LIST(weightExpList))'
COMMENT 'Computes the weighted product of non-missing values in a data set (expList)';

/**
* Register function quartile
* There is only one function with this name, so no need to specify the signature
*/

```

```

CREATE MACRO '@JQTILE'(SINGLE, GROUP)
AS '@_JQTILE(@@1, @@SH1)'
SPEC '@JQTILE(quart,expList)'
COMMENT 'Computes the specified (quart) quartile of a data set (expList)';

CREATE FUNCTION '@_JQTILE'
AS 'com.hyperion.essbase.calculator.Statistics.quartile';

/**
 * Register function rank
 * There is only one function with this name, so no need to specify the signature
 */
CREATE MACRO '@JRANK'(SINGLE, GROUP)
AS '@_JRANK(@@1, @@SH1)'
SPEC '@JRANK(value,expList)'
COMMENT 'Computes the rank of a value in a data set (expList)';

CREATE FUNCTION '@_JRANK'
AS 'com.hyperion.essbase.calculator.Statistics.rank';

/**
 * Register function skewness
 */
CREATE MACRO '@JSKEW'(GROUP)
AS '@_JSKEW(@@S)'
SPEC '@JSKEW(expList)'
COMMENT 'Computes the skewness of a data set (expList)';

CREATE FUNCTION '@JSKEW'
AS 'com.hyperion.essbase.calculator.Statistics.skew(double [])';

/**
 * Register function weighted skewness
 */
CREATE FUNCTION '@JSKEWW'
AS 'com.hyperion.essbase.calculator.Statistics.skew(double [],double [])'
SPEC '@JSKEWW(@LIST(expList), @LIST(weightExpList))'
COMMENT 'Computes the weighted skewness of a data set (expList)';

/**
 * Register function stdev
 */
CREATE FUNCTION '@JSTDEV'(GROUP)
AS '@_JSTDEV(@@S)'
SPEC '@JSTDEV(expList)'
COMMENT 'Computes the standard deviation of non-missing values in a data set (expList)';

CREATE FUNCTION '@_JSTDEV'
AS 'com.hyperion.essbase.calculator.Statistics.stdev(double [])';

/**
 * Register function weighted stdev
 */

```

```

CREATE FUNCTION '@JSTDEVW'
AS 'com.hyperion.essbase.calculator.Statistics.stdev(double [],double [])'
SPEC '@JSTDEVW(@LIST(expList), @LIST(weightExpList))'
COMMENT 'Computes the weighted standard deviation of non-missing values in a data set
(expList)';

```

```

/**
 * Register functions stdev and weighted stdev with a skip instruction.
 * These functions will be used through macros, so no spec/comment specified.
 * Since these functions will not be used directly, the names start with '@_'.
 */

```

```

CREATE FUNCTION '@_JSTDEVWS'
AS 'com.hyperion.essbase.calculator.Statistics.stdev(int,double [])';
CREATE FUNCTION '@_JSTDEVWS'
AS 'com.hyperion.essbase.calculator.Statistics.stdev(int,double [],double [])';

```

```

/**
 * Register macro for stdev with a skip instruction
 */

```

```

CREATE MACRO '@JSTDEVWS' (SINGLE, GROUP)
AS
'@@IFSTRCMP (@@1, SKIPNONE)
  @_JSTDEVWS (0, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPMISSING)
    @_JSTDEVWS (1, @@2)
  @@ELSE
    @@IFSTRCMP (@@1, SKIPZERO)
      @_JSTDEVWS (2, @@2)
    @@ELSE
      @@IFSTRCMP (@@1, SKIPBOTH)
        @_JSTDEVWS (3, @@2)
      @@ELSE
        @@ERROR (@@L1, @_INVALIDSKIP)
      @@ENDIF
    @@ENDIF
  @@ENDIF
@@ENDIF'
SPEC '@JSTDEVWS(SKIPNONE|SKIPZERO|SKIPMISSING|SKIPBOTH, expList)'
COMMENT 'Computes the standard deviation value of a data set (expList) with skip
instructions';

```

```

/**
 * Register macro for weighted standard deviation with a skip instruction
 */

```

```

CREATE MACRO '@JSTDEVWS' (SINGLE, SINGLE, SINGLE)
AS
'@@IFSTRCMP (@@1, SKIPNONE)
  @_JSTDEVWS (0, @@2, @@3)
@@ELSE
  @@IFSTRCMP (@@1, SKIPMISSING)
    @_JSTDEVWS (1, @@2, @@3)
  @@ELSE
    @@IFSTRCMP (@@1, SKIPZERO)

```

```

        @_JSTDEVWS (2, @@2, @@3)
    @@ELSE
        @@IFSTRCMP (@@1, SKIPBOTH)
            @_JSTDEVWS (3, @@2, @@3)
        @@ELSE
            @@ERROR (@@L1, @_INVALIDSKIP)
        @@ENDIF
    @@ENDIF
@@ENDIF
@@ENDIF
@@ENDIF'
SPEC '_JSTDEVWS(SKIPNONE|SKIPZERO|SKIPMISSING|SKIPBOTH, expList, weightExpList)'
COMMENT 'Computes the weighted standard deviation value of a data set (expList) with
skip instructions';

/**
 * Register function stdevp
 */
CREATE MACRO '@JSTDEVP'(GROUP)
AS '@_JSTDEVP(@@S)'
SPEC '@JSTDEVP(expList)'
COMMENT 'Computes the standard deviation(p) of non-missing values in a data set
(expList)';

CREATE FUNCTION '@JSTDEVP'
AS 'com.hyperion.essbase.calculator.Statistics.stdevp(double [])';

/**
 * Register function weighted stdevp
 */
CREATE FUNCTION '@JSTDEVPW'
AS 'com.hyperion.essbase.calculator.Statistics.stdevp(double [],double [])'
SPEC '@JSTDEVPW(@LIST(expList), @LIST(weightExpList))'
COMMENT 'Computes the weighted standard deviation(p) of non-missing values in a data set
(expList)';

/**
 * Register functions stdevp and weighted stdevp with a skip instruction.
 * These functions will be used through macros, so no spec/comment specified.
 * Since these functions will not be used directly, the names start with '@_'.
 */
CREATE FUNCTION '@_JSTDEVPS'
AS 'com.hyperion.essbase.calculator.Statistics.stdevp(int,double [])';
CREATE FUNCTION '@_JSTDEVPWS'
AS 'com.hyperion.essbase.calculator.Statistics.stdevp(int,double [],double [])';

/**
 * Register macro for stdevp with a skip instruction
 */
CREATE MACRO '@JSTDEVPS'(SINGLE,GROUP)
AS
'@@IFSTRCMP (@@1, SKIPNONE)
    @_JSTDEVPS (0, @@2)
@@ELSE

```

```

@@IFSTRCMP (@@1, SKIPMISSING)
  @_JSTDEVPS (1, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPZERO)
    @_JSTDEVPS (2, @@2)
  @@ELSE
    @@IFSTRCMP (@@1, SKIPBOTH)
      @_JSTDEVPS (3, @@2)
    @@ELSE
      @@ERROR (@@L1, @_INVALIDSKIP)
    @@ENDIF
  @@ENDIF
@@ENDIF
@@ENDIF'
SPEC '@JSTDEVPS(SKIPNONE|SKIPZERO|SKIPMISSING|SKIPBOTH, expList)'
COMMENT 'Computes the standard deviation(p) value of a data set (expList) with skip
instructions';

/**
 * Register macro for weighted stdevp with a skip instruction
 */
CREATE MACRO '@JSTDEVPWS' (SINGLE,SINGLE,SINGLE)
AS
'@@IFSTRCMP (@@1, SKIPNONE)
  @_JSTDEVPWS (0, @@2, @@3)
@@ELSE
  @@IFSTRCMP (@@1, SKIPMISSING)
    @_JSTDEVPWS (1, @@2, @@3)
  @@ELSE
    @@IFSTRCMP (@@1, SKIPZERO)
      @_JSTDEVPWS (2, @@2, @@3)
    @@ELSE
      @@IFSTRCMP (@@1, SKIPBOTH)
        @_JSTDEVPS (3, @@2, @@3)
      @@ELSE
        @@ERROR (@@L1, @_INVALIDSKIP)
      @@ENDIF
    @@ENDIF
  @@ENDIF
@@ENDIF'
SPEC '@JSTDEVPWS(SKIPNONE|SKIPZERO|SKIPMISSING|SKIPBOTH, expList, weightExpList)'
COMMENT 'Computes the weighted standard deviation(p) value of a data set (expList) with
skip instructions';

/**
 * Register function sum
 */
CREATE MACRO '@JSUM' (GROUP)
AS '@_JSUM(@@S)'
SPEC '@JSUM(expList)'
COMMENT 'Computes the sum of a data set (expList)';

CREATE FUNCTION '@_JSUM'
AS 'com.hyperion.essbase.calculator.Statistics.sum(double [])';

```



```

/**
 * Register function weighted SUM
 */
CREATE FUNCTION '@JSUMW'
AS 'com.hyperion.essbase.calculator.Statistics.sum(double [],double [])'
SPEC '@JSUMW(@LIST(expList), @LIST(weightExpList))'
COMMENT 'Computes the weighted sum of a data set (expList)';

/**
 * Register function var
 */
CREATE MACRO '@JVAR'(GROUP)
AS '@_JVAR(@@S)'
SPEC '@JVAR(expList)'
COMMENT 'Computes the variance of non-missing values in a data set (expList)';

CREATE FUNCTION '@_JVAR'
AS 'com.hyperion.essbase.calculator.Statistics.var(double [])';

/**
 * Register function weighted var
 */
CREATE FUNCTION '@JVARW'
AS 'com.hyperion.essbase.calculator.Statistics.var(double [],double [])'
SPEC '@JVARW(@LIST(expList), @LIST(weightExpList))'
COMMENT 'Computes the weighted variance of non-missing values in a data set (expList)';

/**
 * Register functions var and weighted var with a skip instruction.
 * These functions will be used through macros, so no spec/comment specified.
 * Since these functions will not be used directly, the names start with '@_'.
 */
CREATE FUNCTION '@_JVARs'
AS 'com.hyperion.essbase.calculator.Statistics.var(int,double [])';
CREATE FUNCTION '@_JVARWS'
AS 'com.hyperion.essbase.calculator.Statistics.var(int,double [],double [])';

/**
 * Register macro for var with a skip instruction
 */
CREATE MACRO '@JVARs'(SINGLE,GROUP)
AS
'@@IFSTRCMP (@@1, SKIPNONE)
  @_JVARs (0, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPMISSING)
  @_JVARs (1, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPZERO)
  @_JVARs (2, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPBOTH)
  @_JVARs (3, @@2)

```

```

        @@ELSE
            @@ERROR (@@L1, @_INVALIDSKIP)
        @@ENDIF
    @@ENDIF
    @@ENDIF
    @@ENDIF'
SPEC '@JVARWS(SKIPNONE|SKIPZERO|SKIPMISSING|SKIPBOTH, expList)'
COMMENT 'Computes the variance value of a data set (expList) with skip instructions';

/**
 * Register macro for weighted variance with a skip instruction
 */
CREATE MACRO '@JVARWS' (SINGLE, SINGLE, SINGLE)
AS
'@@IFSTRCMP (@@1, SKIPNONE)
    @_JVARWS (0, @@2, @@3)
@@ELSE
    @@IFSTRCMP (@@1, SKIPMISSING)
        @_JVARWS (1, @@2, @@3)
    @@ELSE
        @@IFSTRCMP (@@1, SKIPZERO)
            @_JVARWS (2, @@2, @@3)
        @@ELSE
            @@IFSTRCMP (@@1, SKIPBOTH)
                @_JVARWS (3, @@2, @@3)
            @@ELSE
                @@ERROR (@@L1, @_INVALIDSKIP)
            @@ENDIF
        @@ENDIF
    @@ENDIF
    @@ENDIF'
SPEC '@JVARWS(SKIPNONE|SKIPZERO|SKIPMISSING|SKIPBOTH, expList, weightExpList)'
COMMENT 'Computes the weighted variance value of a data set (expList) with skip
instructions';

/**
 * Register function varp
 */
CREATE MACRO '@JVARP' (GROUP)
AS '@_JVARP(@@S)'
SPEC '@JVARP(expList)'
COMMENT 'Computes the variance(p) of non-missing values in a data set (expList)';

CREATE FUNCTION '@_JVARP'
AS 'com.hyperion.essbase.calculator.Statistics.varp(double [])';

/**
 * Register function weighted varp
 */
CREATE FUNCTION '@JVARPW'
AS 'com.hyperion.essbase.calculator.Statistics.varp(double [], double [])'
SPEC '@JVARPW(@LIST(expList), @LIST(weightExpList))'
COMMENT 'Computes the weighted variance(p) of non-missing values in a data set
(expList)';

```

```

/**
 * Register functions varp and weighted varp with a skip instruction.
 * These functions will be used through macros, so no spec/comment specified.
 * Since these functions will not be used directly, the names start with '@_'.
 */
CREATE FUNCTION '@_JVARPS'
AS 'com.hyperion.essbase.calculator.Statistics.varp(int,double [])';
CREATE FUNCTION '@_JVARPWS'
AS 'com.hyperion.essbase.calculator.Statistics.varp(int,double [],double [])';

/**
 * Register macro for varp with a skip instruction
 */
CREATE MACRO '@JVARPS' (SINGLE,GROUP)
AS
'@@IFSTRCMP (@@1, SKIPNONE)
  @_JVARPS (0, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPMISSING)
  @_JVARPS (1, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPZERO)
  @_JVARPS (2, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPBOTH)
  @_JVARPS (3, @@2)
@@ELSE
  @@ERROR (@@L1, @_INVALIDSKIP)
@@ENDIF
@@ENDIF
@@ENDIF'
SPEC '@JVARPS(SKIPNONE|SKIPZERO|SKIPMISSING|SKIPBOTH, expList)'
COMMENT 'Computes the variance(p) value of a data set (expList) with skip instructions';

/**
 * Register macro for weighted varp with a skip instruction
 */
CREATE MACRO '@JVARPWS' (SINGLE,SINGLE,SINGLE)
AS
'@@IFSTRCMP (@@1, SKIPNONE)
  @_JVARPWS (0, @@2, @@3)
@@ELSE
  @@IFSTRCMP (@@1, SKIPMISSING)
  @_JVARPWS (1, @@2, @@3)
@@ELSE
  @@IFSTRCMP (@@1, SKIPZERO)
  @_JVARPWS (2, @@2, @@3)
@@ELSE
  @@IFSTRCMP (@@1, SKIPBOTH)
  @_JVARPWS (3, @@2, @@3)
@@ELSE
  @@ERROR (@@L1, @_INVALIDSKIP)
@@ENDIF
@@ENDIF

```

```

    @@ENDIF
  @@ENDIF
  @@ENDIF '
SPEC ' @JVARPWS (SKIPNONE|SKIPZERO|SKIPMISSING|SKIPBOTH, expList, weightExpList) '
COMMENT 'Computes the weighted variance(p) value of a data set (expList) with skip
instructions';

```

Custom-Defined Macros

Custom-defined macros enable you to combine Essbase [calculation functions](#) into a single function, called a macro. Custom-defined macros can also include special [directives](#), variables, and other macros. After you create macros, they can be used in formulas and calculation scripts just like native Essbase calculation functions.

Note: Custom-defined macros cannot include [calculation commands](#).

Topics that discuss custom-defined macros:

- [“Custom-Defined Macro Input Parameters” on page 292](#)
- [“Using Argument Values in Macro Definitions” on page 294](#)
- [“Directives Used in Custom-Defined Macros” on page 295](#)
- [“Macro Reference” on page 295](#)

For information about creating custom-defined macros, see the MaxL DDL [Create Macro](#) statement. For more information about custom-defined macros, see the *Oracle Essbase Database Administrator's Guide*.

Custom-Defined Macro Input Parameters

When creating a macro, you can define how many and what kind of arguments are passed into the macro. Specifying the argument set (also known as the signature) for a macro is optional, but specifying it can make the macro easier to use and prevent usage errors.

The argument set is specified as part of the macro name when you create a macro with the [Create Macro](#) MaxL statement. In the following macro name, the argument set is enclosed in parentheses:

```
@SUMRANGE(single, group)
```

The preceding macro signature indicates that this macro requires two arguments: *single*, which represents one input parameter, and *group*, which represents a list of input parameters. These macro arguments do not represent a specific data type (such as a boolean, double, or string); instead, they only indicate *how many* arguments are accepted by the macro.

Arguments are specified in a comma-delimited list (*argument1, argument2, ... argumentX*) as part of the macro name when the macro is created. Arguments can be specified using the following keywords, which tell the macro processor how to check the arguments for a macro:

Argument	Description
SINGLE	A single argument
GROUP	A list of arguments. Any argument following GROUP is ignored.
OPTIONAL	A single argument that is not required
OPTIONAL_GROUP	A list of arguments that is not required. Any argument following OPTIONAL_GROUP is ignored.
ANY	No checking of arguments. Any argument following ANY is ignored.

In the macro presented previously, the following sets of arguments are valid:

```
@SUMRANGE(Profit, @CHILDREN(East))
@SUMRANGE(Profit, "New York", "New Jersey", Connecticut)
@SUMRANGE(Sales, @DESCENDANTS(Product))
```

The following table shows examples of how the macro processor interprets arguments for macros with different signatures given different input parameters. The definition of the example macro is:

```
create macro SUM3(argument1, argument2, argument3) as '@@1 + @@2 + @@3';
```

Macro with Signature of SUM3(signature)	Result when given input of SUM3(X,Y)	Result when given input of SUM3(X,Y,Z)	Result when given input of SUM3(X,Y,Z,T)
SUM3(SINGLE, SINGLE, SINGLE)	Error (wrong number of arguments)	X+Y+Z	Error (wrong number of arguments)
SUM3(SINGLE, SINGLE, GROUP)	Error (wrong number of arguments)	X+Y+Z	X+Y+@LIST(Z,T)
SUM3(SINGLE, SINGLE, OPTIONAL_GROUP)	X+Y+@_NULL	X+Y+Z	X+Y+@LIST(Z,T)
SUM3(SINGLE, SINGLE, OPTIONAL)	X+Y+@_NULL	X+Y+Z	Error (wrong number of arguments)
SUM3(SINGLE, SINGLE, ANY)	X+Y+@_NULL	X+Y+Z	X+Y+Z
SUM3(SINGLE, ANY)	X+Y+	X+Y+Z	X+Y+Z
SUM3(SINGLE, GROUP)	X+Y+	X+@LIST(Y,Z)+	X+@LIST(Y,Z,T)+
SUM3(ANY)	X+Y+	X+Y+Z	X+Y+Z

As noted previously, specification of arguments in the macro name only restricts the number of arguments that are accepted by the macro and does not restrict the data types that may be passed into the macro. Arguments in the Essbase calculator language can represent any of the following data types:

Data Type	Description
Number	A single, double precision, floating point type number, which can have a special value, #MISSING, or an array of these numbers
Boolean	A single three-valued variable with the possible values, TRUE, FALSE, and #MISSING, or an array of these variables
Member	A single database outline member, cross-member combination, or an array of members
String	A string variable type, or an array of these strings

When developing macros, you should consider the types of data that can be passed into macros to avoid errors in calculation.

Using Argument Values in Macro Definitions

Specifying an argument set for a custom-defined macro is only part of creating a macro. You must use the argument values in the macro expansion, which defines what functions the macro performs. Two types of argument variables can be used in a macro definition: numbered argument variables and argument variable shortcuts.

Using Numbered Argument Variables

In a macro definition, argument variables can be referenced by the order in which they appear in the macro signature. Consider the following example macro signature with three argument variables:

```
SUM3(single, single, group)
```

To use the input from this function in the macro definition, you reference the arguments using the argument variables @@1 for the first input parameter, @@2 for the second input parameter, and @@3 for the third input parameter. Thus, using the macro in the preceding example and providing the following input,

```
SUM3("New York", "New Jersey", @CHILDREN(Products));
```

results in the macro variables being set to the following values:

```
@@1 = "New York"@@2 = "New Jersey"@@3 = @CHILDREN(Products)
```

Use of the optional argument in the macro signature has no effect on which macro variable represents which incoming argument; for example, the input,

```
Macro signature: SUM3(single, optional, group)
Macro input: SUM3("New York", , @CHILDREN(Products));
```

results in the macro variables being set to the following values:

```
@@1 = "New York"@@2 = @_NULL@@3 = @CHILDREN(Products)
```

Using Argument Variable Shortcuts

You can represent sets of arguments with the variable shortcuts @@S and @@SHx. These shortcuts enable you to specify a set of arguments with one variable, rather than listing a set of numbered

variables. Using input from the preceding example, the `@@S` variable would be set to the following value:

```
@@S = "New York", @NULL, @CHILDREN(Products)
```

Argument variables and shortcuts for custom-defined macros can be used in any order within a macro definition and can be repeated in a macro.

Directives Used in Custom-Defined Macros

Custom-defined macros can include [calculation functions](#), but cannot include [calculation commands](#).

In addition to the calculation functions, custom-defined macros can include special directives that are available only for macros. These directives are categorized as follows:

Variable handling

- “`@@x`” on page 295
- “`@@S`” on page 296
- “`@@SHx`” on page 297

Error handling

- “`@@ERROR`” on page 297
- “`@@Lx`” on page 298

Conditionals

- “`@@IFSTRCMP`” on page 299
- “`@@ELSE`” on page 300
- “`@@ENDIF`” on page 301

Macro Reference

The following topics describe the directives.

`@@x`

The `@@x` statement is a variable representing an input argument for a macro. The number x is the number of the argument in the signature of the macro. So, `@@1` represents the first input argument, `@@2` represents the second input argument, and so on.

Syntax

```
@@x
```

Where x is the number of an argument in the signature of the macro.

Notes

- Each @@x input argument variable can be used multiple times within a macro expansion.
- The @@x argument variable can also be used with the @@S and @@SHx argument variables within a macro expansion.
- The meaning of @@x argument variables does not change if an optional variable is not provided; for example, given the following macro signature,
create macro Sample.'@ADD'(single, optional, single) as '(@@1 + @@2 + @@3)';

and the following input parameters,

```
@ADD("New York", , Connecticut);
```

the argument variables would be set to these values:

```
@@1 = "New York"  
@@2 = @_NULL  
@@3 = Connecticut
```

Example

The following example shows a create statement for a macro with three input arguments that are added.

```
create macro Sample.'@SUM3'(single, single, single) as '(@@1 + @@2 + @@3)';
```

See Also

- [“@@S” on page 296](#)
- [“@@SHx” on page 297](#)

@@S

The @@S statement is a variable representing all input arguments for a macro.

Syntax

```
@@S
```

Notes

- The @@S input argument variable can be used multiple times within a macro expansion.
- The @@S input argument variable can also be used with the @@x and @@SHx argument variables within a macro expansion.

Example

The following example shows a macro that divides the sum of all arguments by the sum of the first two arguments.

```
create macro Sample.'@DIVIDE'(single, single, optional_group)  
as '@SUM(@@S)/(@@1 + @@2)';
```


See Also

- [“@@x” on page 295](#)
- [“@@SHx” on page 297](#)

@@SHx

The @@SHx statement represents a subset of all arguments starting with position *x* and including the rest of the arguments for the macro.

Syntax

@@Sx

Where *x* is the number of an argument in the signature of the macro, with 0 representing the first position, 1 representing the second position, and so on.

Notes

- The @@SHx argument variable can be used multiple times within a macro expansion.
- The @@SHx argument variable can be used with the @@x and @@S argument variables within a macro expansion.

Example

The following example shows a macro that multiplies the first arguments together and adds them to the sum of the remaining arguments.

```
create macro Sample.'@MULTANDSUM'(single, single, any)
  as '(@@1 * @@2) + @SUM(@@SH2)';
```

See Also

- [“@@x” on page 295](#)
- [“@@S” on page 296](#)

@@ERROR

The @@ERROR command forces the macro processor to stop and report an error.

Syntax

@@ERROR(*lineNumber* , *errorCode*)

Where:

- *lineNumber* is a number representing a line in the calculation script or formula where the macro is used
- *errorCode* is an error code for the error

Notes

The @@Lx command can be used as the first parameter of an @@ERROR statement to identify a line number in a calculation script or formula where the macro is used.

Example

The following example function checks the first input argument for valid values (SKIPNONE, SKIPMISSING, SKIPZERO, SKIPBOTH). If none of these values is found, the macro returns an error, specifying a line number in a calculation script or formula where the macro is used.

```
@@IFSTRCMP (@@1, SKIPNONE)
  @_JAVGS (0, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPMISSING)
    @_JAVGS (1, @@2)
  @@ELSE
    @@IFSTRCMP (@@1, SKIPZERO)
      @_JAVGS (2, @@2)
    @@ELSE
      @@IFSTRCMP (@@1, SKIPBOTH)
        @_JAVGS (3, @@2)
      @@ELSE
        @@ERROR (@@L1, @_INVALIDSKIP)
      @@ENDIF
    @@ENDIF
  @@ENDIF
@@ENDIF
```

See Also

- [“@@Lx” on page 298](#)
- [“@@IFSTRCMP” on page 299](#)

@@Lx

The @@Lx command returns a number representing the line in a calculation script or formula where a macro argument occurs, or the line where the macro name occurs.

Syntax

@@Lx

Where *x* is a number specifying a macro input argument number (1, 2, ... *n*), or the macro name, if zero (0) is specified.

Notes

The @@Lx command can be used only as the first parameter of an @@ERROR statement to identify a line number for an error in a calculation script or formula.

Example

The following example macro checks the first input argument for valid values (SKIPNONE, SKIPMISSING, SKIPZERO, SKIPBOTH). If none of these values is found, the macro returns an error, specifying a line number in a calculation script or formula where the macro is used. The line number is specified using the @@L1 statement, which returns 2, the number of the line in the calculation script or formula where the first parameter of the macro occurs.

```
Calculation script using macro @AVGS
1: "Average_Revenue" = @AVGS(
2:                     SKIPNONE,
3:                     @CHILDREN(YrlyRevenue)
4:                     );
```

@AVGS macro definition:

```
@@IFSTRCMP (@@1, SKIPNONE)
  @_JAVGS (0, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPMISSING)
    @_JAVGS (1, @@2)
  @@ELSE
    @@IFSTRCMP (@@1, SKIPZERO)
      @_JAVGS (2, @@2)
    @@ELSE
      @@IFSTRCMP (@@1, SKIPBOTH)
        @_JAVGS (3, @@2)
      @@ELSE
        @@ERROR (@@L1, @_INVALIDSKIP)
      @@ENDIF
    @@ENDIF
  @@ENDIF
@@ENDIF
```

See Also

[“@@ERROR” on page 297](#)

@@IFSTRCMP

The @@IFSTRCMP command compares a macro input parameter to a string. If the input parameters match, the macro statements following the command are processed. Otherwise, the statements following @@ELSE are processed.

Syntax

```
@@IFSTRCMP( @@x , token ) statement @@ELSE... [statement]
@@ENDIF
```

Where:

- @@x is a variable representing a macro argument
- token is a string to be compared to the macro argument

- *statementis* operations to be performed depending on the results of the test

Notes

The `@@IFSTRCMP` statement block must use the `@@ELSE` statement as part of its decision syntax. You do not have to include a statement after `@@ELSE`.

Example

```
@@IFSTRCMP (@@2, @_NULL)
  @@1
@@ELSE
  (@@1 + @@2)
@@ENDIF
```

This test checks to see if the second macro argument is blank. If it is, then only the first argument is used. If the second argument is not blank, then the two arguments are added.

See Also

- [“@@ELSE” on page 300](#)
- [“@@ENDIF” on page 301](#)

@@ELSE

The `@@ELSE` command designates a conditional action to be performed in an `@@IFSTRCMP` statement. All actions placed after the `@@ELSE` in an `@@IFSTRCMP` statement are performed only if the strings compared in the `@@IFSTRCMP` statement do not match.

Syntax

```
@@ELSE...statement [ ...statement ] @@ENDIF
```

Where *statementis* operations to be performed depending on the results of the test.

Notes

- The `@@ELSE` statement can only be used in conjunction with an `@@IFSTRCMP` statement.
- All `@@IFSTRCMP` statements must be ended with `@@ENDIF` statements.

Example

```
@@IFSTRCMP (@@2, @_NULL)
  @@1
@@ELSE
  (@@1 + @@2)
@@ENDIF
```

This test checks to see if the second macro argument is blank. If it is, then only the first argument is used. If the second argument is not blank, then the two arguments are added.

See Also

- “[@@IFSTRCMP](#)” on page 299
- “[@@ENDIF](#)” on page 301

@@ENDIF

The `@@ENDIF` command marks the end of an `@@IFSTRCMP` command sequence. The `@@ENDIF` command can be used only in conjunction with the `@@IFSTRCMP` statement.

Syntax

```
@@ENDIF
```

Notes

- You must supply an `@@ENDIF` statement for every `@@IFSTRCMP` statement in your macro. If you do not supply the required `@@ENDIF` statements, your formula or calculation script does not verify.
- If you are using an IF statement nested within another IF statement, end each IF with an `ENDIF`, as in the following example:

```
@@IFSTRCMP (@@1, SKIPNONE)
  @_JAVGS (0, @@2)
@@ELSE
  @@IFSTRCMP (@@1, SKIPMISSING)
    @_JAVGS (1, @@2)
  @@ELSE
    @@IFSTRCMP (@@1, SKIPZERO)
      @_JAVGS (2, @@2)
    @@ELSE
      @@IFSTRCMP (@@1, SKIPBOTH)
        @_JAVGS (3, @@2)
      @@ELSE
        @@ERROR (@@L1, @_INVALIDSKIP)
      @@ENDIF
    @@ENDIF
  @@ENDIF
@@ENDIF
```

- All `@@IFSTRCMP` statements must be ended with `@@ENDIF` statements.

Example

```
@@IFSTRCMP (@@2, @_NULL)
  @@1
@@ELSE
  (@@1 + @@2)
@@ENDIF
```

This test checks to see if the second macro argument is blank. If it is, then only the first argument is used. If the second argument is not blank, then the two arguments are added.

See Also

- [“@@IFSTRCMP” on page 299](#)
- [“@@ELSE” on page 300](#)

3

Calculation Commands

In This Chapter

Calculation Commands Overview	303
Calculation Operators	303
Calculation Command Groups	305
Calculation Command Reference	308

Calculation Commands Overview

You use calculation scripts to create calculations that differ from those defined in the database outline. Calculation scripts enable development of custom operations to supplement the built-in calculation of the database outline.

Calculation commands are the elements of calculation scripts that instruct Essbase in the calculation rules to be used. You create calculation scripts using the Calculation Script Editor. Within the Calculation Script Editor, a dialog box is available that allows you to paste functions while you develop formulas. For more information, see the *Oracle Essbase Administration Services Online Help*.

When a database is created, a default calculation script is set to "calculate all", which means that it will calculate all dimensions based on the database outline's hierarchical relationships and formulas.

You can override this default script by using a custom script. You can use the custom script(s) temporarily or permanently, without altering the default script. In the custom script, you can refer to calculation rules defined in the database outline or you can specify custom formulas, calculation formats, and calculation orders.

A calculation script contains a series of calculation commands. The order of the commands defines the execution order of the calculation.

Calculation Operators

Calculation operators (mathematical, conditional and logical, and cross-dimensional) define equations for member formulas and calc scripts.

- [“Mathematical Operators” on page 304](#)
- [“Conditional and Logical Operators” on page 304](#)

- [“Cross-Dimensional Operator” on page 305](#)

Mathematical Operators

Mathematical operators perform common arithmetic operations.

Operator	Description
+	Adds
-	Subtracts
*	Multiplies
/	Divides
%	Evaluates percentage, for example: <i>Member1%Member2</i> evaluates <i>Member1</i> as a percentage of <i>Member2</i> .
()	Controls the order of calculations and nests equations and formulas

Conditional and Logical Operators

Conditional operators build logical condition into calculations.

Operator	Description
IF ELSE ELSEIF ENDIF	Tests conditions and calculates a formula based on the success or failure of the test
>	Data value is greater than
>=	Data value is greater than or equal to
<	Data value is less than
<=	Data value is less than or equal to
= =	If data value is equal to
< > or !=	Data value is not equal to
AND	Logical AND linking operator for multiple value tests. Result is TRUE if both conditions are TRUE. Otherwise the result is FALSE. ¹
OR	Logical OR linking operator for multiple value tests. Result is TRUE if either condition is TRUE. Otherwise the result is FALSE. ²
NOT	Logical NOT operator. Result is TRUE if condition is FALSE. Result is FALSE if condition is TRUE. ³

¹The logical constants TRUE and FALSE are interpreted as 1 (TRUE) and 0 (FALSE) where appropriate.

²The logical constants TRUE and FALSE are interpreted as 1 (TRUE) and 0 (FALSE) where appropriate.

³The logical constants TRUE and FALSE are interpreted as 1 (TRUE) and 0 (FALSE) where appropriate.

Cross-Dimensional Operator

The cross-dimensional operator points to data values of specific member combinations. It is created with a hyphen (-) and a right angle bracket (>), with no space between them: ->

Calculation Command Groups

This section lists calculation commands grouped by type:

- [“Conditional Commands” on page 305](#)
- [“Control Flow Commands” on page 306](#)
- [“Data Declaration Commands” on page 306](#)
- [“Functional Commands” on page 306](#)
- [“Member Formulas” on page 307](#)

Conditional Commands

Conditional commands control the flow of events in formulas. You can control which formulas are executed within a calculation, test conditions, and calculate a formula based on the result of the test.

- [IF](#)
- [ENDIF](#)
- [ELSE](#)
- [ELSEIF](#)

When you use an IF statement as part of a member formula in a calc script, you need to:

- Associate it with a single member
- Enclose it in parentheses

For example:

```
Profit (IF (Sales > 100)
    Profit = (Sales - COGS) * 2;
ELSE
    Profit = (Sales - COGS) * 1.5;
ENDIF);
```

Essbase cycles through the database, performing the following calculations:

1. The IF statement checks to see if the value of Sales for the current member combination is greater than 100.
2. If Sales is greater than 100, Essbase subtracts the value in COGS from the value in Sales, multiplies it by 2, and places the result in Profit.

3. If Sales is less than, or equal to 100, Essbase subtracts the value in COGS from the value in Sales, multiplies it by 1.5, and places the result in Profit.

The entire IF fixend.htm ENDIF statement is enclosed in parentheses and associated with the Profit member, `Profit (IF(fixend.htm) fixend.htm)`.

Control Flow Commands

Control Flow commands are used to iterate a set of commands or to restrict the commands' effect to a subset (partition) database. They control the flow of a calculation script. The [FIX...ENDFIX](#) and [EXCLUDE...ENDEXCLUDE](#) commands restrict calculations to specified members. The [LOOP...ENDLOOP](#) command enables repetition.

Data Declaration Commands

These commands are used to declare and set the initial values of temporary variables. The values stored in a variable are not returned in queries, because they only exist while the calculation script is being processed. If you want to report these values, you need to create members within the database outline, or assign the values from the variables into existing members.

- [ARRAY](#)
- [VAR](#)

Functional Commands

Functional commands are used to perform operations such as calculation, data copying, exporting data, clearing data, and Currency Conversion.

- [AGG](#)
- [CALC ALL](#)
- [CALC AVERAGE](#)
- [CALC DIM](#)
- [CALC FIRST](#)
- [CALC LAST](#)
- [CALC TWOPASS](#)
- [CCONV](#)
- [CLEARBLOCK](#)
- [CLEARCCTRACK](#)
- [CLEARDATA](#)
- [DATACOPY](#)
- [DATAEXPORT](#)

- DATAEXPORTCOND
- DATAIMPORTBIN
- SET DATAEXPORTOPTIONS
- SET DATAIMPORTIGNORETIMESTAMP
- SET AGGMISSG
- SET CACHE
- SET CTRACKCALC
- SET CLEARUPDATESTATUS
- SET FRMLBOTTOMUP
- SET FRMLRTDYNAMIC
- SET LOCKBLOCK
- SET MSG
- SET NOTICE
- SET REMOTECALC
- SET UPDATECALC
- SET UPTOLOCAL

Member Formulas

Member Formulas are used to calculate the default outline format on a custom formula within the script. As with formulas in the database outline, a formula in a calculation script defines mathematical relationships between database members. For example, the following expressions are valid within a calculation script:

```
"Profit_%";
```

Specifying a member name with a formula defined in the outline calculates the member using its formula.

```
Expenses = Payroll + Marketing;
```

The above formula expresses a simple mathematical relationship, which is used in place of the database outline formula on the Expenses member.

Interdependent Member Formulas

Essbase optimizes calculation performance by calculating formulas for a range of members in the same dimension. However, some formulas require values from members of the same dimension. A good example is that of cash flow, in which the opening inventory is dependent on the closing inventory from the previous month.

For examples of interdependent formulas, see the *Oracle Essbase Database Administrator's Guide*.

When you use an interdependent formula in a calc script, the same rules apply as for the IF statement. You need to:

- Associate the formula with a single member
- Enclose the formula in parentheses

If you place the following interdependent formula in a calc script, you construct it as follows:

```
"Opening Inventory" (IF(NOT @ISMBR (Jan) "Opening Inventory" =
@PRIOR("Ending Inventory"));
ENDIF;
"Ending Inventory" = "Opening Inventory" - Sales + Additions;)
```

The entire formula is enclosed in parentheses and associated with the Opening Inventory member, "Opening Inventory" (IF(fixend.htm)...).

Calculation Command Reference

Consult the Contents pane for a categorical list of calculation commands.

&	DATAIMPORTBIN	SET CREATEBLOCKONEQ
AGG	ELSE	SET DATAEXPORTOPTIONS
ARRAY	ELSEIF	SET DATAIMPORTIGNORETIMESTAMP
CALC ALL	ENDIF	SET EMPTYMEMBERSETS
CALC AVERAGE	EXCLUDE...ENDEXCLUDE	SET FRMLBOTTOMUP
CALC DIM	FIX...ENDFIX	SET FRMLRTDYNAMIC
CALC FIRST	IF	SET LOCKBLOCK
CALC LAST	LOOP...ENDLOOP	SET MSG
CALC TWOPASS	SET AGGMISG	SET NOTICE
CCONV	SET CACHE	SET REMOTECALC
CLEARBLOCK	SET CALCPARALLEL	SET SCAPERSPECTIVE
CLEARCCTRACK	SET CALCTASKDIMS	SET UPDATECALC
CLEARDATA	SET CCTRACKCALC	SET UPTOLOCAL
DATACOPY	SET CLEARUPDATESTATUS	VAR
DATAEXPORT	SET COPYMISSINGBLOCK	
DATAEXPORTCOND	SET CREATENONMISSINGBLK	

&

Prefaces a substitution variable in a calculation script.

Syntax

```
&variableName;
```

Parameter	Description
-----------	-------------

variableName	The name of the substitution variable set on the database.
--------------	------------------------------------------------------------

Notes

Essbase treats strings beginning with & as substitution variables, replacing them with values before parsing the calculation script.

Example

```
&CurQtr;
```

becomes

```
Qtr1;
```

if substitution variable &CurQtr has the value "Qtr1".

AGG

Consolidates database values. This command ignores all member formulas, consolidating only parent/child relationships.

The AGG command performs a limited set of high-speed consolidations. Although AGG is faster than the CALC commands when calculating sparse dimensions, it cannot calculate formulas; it can only perform aggregations based on the database structure. AGG aggregates a list of *sparse* dimensions based on the hierarchy defined in the database outline. If a member has a formula, it is ignored, and the result does not match the relationship defined by the database outline.

If you want to aggregate a dimension that contains formulas:

1. Calculate any members that are "leaf" members (that is, level 0).
2. Aggregate the dimension, using the AGG command.
3. Calculate all other members with formulas that have not been calculated yet.

Syntax

```
AGG (dimList);
```

Parameter	Description
-----------	-------------

dimList	Name of a dimension or comma-separated list of dimensions.
---------	------------------------------------------------------------

Notes

- AGG only works with *sparse* dimensions.

- When a dimension contains fewer than six consolidation levels, AGG is typically faster than CALC. Conversely, the CALC command is usually faster on dimensions with six or more levels.
- AGG follows the rules for any defined FIX command.

Example

```
AGG(Market);
AGG(Product,Market,Scenario);
```

See Also

- [CALC ALL](#)
- [CALC DIM](#)
- [SET AGGMISSG](#)

ARRAY

Declares one-dimensional array variables.

Syntax

```
ARRAY arrayVariableName [dimName] = { constList};
```

Parameter	Description
<i>arrayVariableName</i>	Comma-delimited list of one or more array variable names.
<i>dimName</i>	Dimension whose size determines the size of the array variable. Surround <i>dimName</i> with brackets [].
<i>constList</i>	Optional list of data values used to initialize the array variable(s). If no initialization is performed, the array variables are set to #MISSING. The order of the values corresponds to the order of the members in the dimension used to define the array.

Notes

- Typically, arrays are used to temporarily store variables as part of a member formula. The variables cease to exist after the calculation script ends. The size of the array variable is determined by the corresponding dimension (e.g., if dimension Period has 12 members, ARRAY Discount[Period] has 12 members). To create multiple arrays simultaneously, separate the array declarations in the ARRAY command with commas, as shown in the Example.
- You can calculate data for an array directly as part of a member formula. As the member formula is processed, each value in the array is assigned as its member is evaluated in the calculation.
- Do not use quotation marks (") in variables; for example:

```
ARRAY "discount"
```

Example

```
ARRAY discount[Scenario];
```

yields an array of 4 entries, with the values 1 through 4 entered in those four entries.

```
ARRAY discount[Scenario] = {1, 2, 3, 4};  
ARRAY discount[Scenario], tmpProduct[Product];
```

yields two arrays:

1. `discount`, corresponding to `Scenario` and containing four members
2. `tmpProduct`, corresponding to `Product` and containing nine members

See Also

- [VAR](#)

CALC ALL

Calculates and aggregates the entire database based on the database outline.

Syntax

```
CALC ALL [EXCEPT DIM (dimList) | MBR (mbrList)];
```

Parameter Description

EXCEPT Defines an exception list of dimensions or members to be excluded from calculation.

DIM Single-dimension specification.

dimList Optional comma-delimited list of dimensions.

MBR Single-member specification.

mbrList Optional comma-delimited list of members, member set functions, or range functions.

Notes

The order in which dimensions are processed depends on their characteristics in the outline. For more information, see "Defining Calculation Order" in the *Oracle Essbase Database Administrator's Guide*.

Example

```
CALC ALL;  
CALC ALL EXCEPT DIM(Product);
```

See Also

- [CALC DIM](#)
- [SET UPDATECALC](#)
- [SET FRMLBOTTOMUP](#)

CALC AVERAGE

Calculates members tagged as time balance Average or Average Non-Missing. All other member calculations are ignored.

Syntax

```
CALC AVERAGE;
```

Notes

This command calculates based on the Accounts dimension; it does not do a Time Series calculation on the Time dimension.

Example

```
CALC AVERAGE;
```

See Also

- [CALC FIRST](#)
- [CALC LAST](#)

CALC DIM

Calculates formulas and aggregations for each member of the specified dimensions.

Syntax

```
CALC DIM (dimList);
```

Parameter Description

dimList Dimension or comma-delimited list of dimensions to be calculated.

Notes

The order in which dimensions are calculated depends on whether they are dense or sparse. Dense dimensions are calculated first, in the order of *dimList*. The sparse dimensions are then calculated in a similar order.

Example

```
CALC DIM(Accounts);
```

```
CALC DIM(Dense1, Sparse1, Sparse2, Dense2);
```

In the above example, the calculation order is: Dense1, Dense2, Sparse1, Sparse2. If your dimensions need to be calculated in a particular order, use separate CALC DIM commands:

```
CALC DIM(Dense1);  
CALC DIM(Sparse1);  
CALC DIM(Sparse2);  
CALC DIM(Dense2);
```

See Also

- [CALC ALL](#)
- [SET UPDATECALC](#)
- [SET CLEARUPDATESTATUS](#)

CALC FIRST

Calculates all members tagged in the database outline as time balance First.

Note: Only members tagged as time balance First are calculated using this command. Other members are ignored.

Syntax

```
CALC FIRST;
```

Notes

This command calculates based on the Accounts dimension; it does not do a Time Series calculation on the Time dimension.

Example

```
CALC FIRST;
```

See Also

- [CALC AVERAGE](#)
- [CALC LAST](#)

CALC LAST

Calculates all members tagged in the database outline as time balance Last.

Note: Only members tagged as time balance Last are calculated using this command. Other members are ignored.

Syntax

```
CALC LAST;
```

Notes

This command calculates based on the Accounts dimension; it does not do a Time Series calculation on the Time dimension.

Example

```
CALC LAST;
```

See Also

- [CALC AVERAGE](#)
- [CALC FIRST](#)

CALC TWOPASS

Calculates all members tagged in the database outline as two-pass. These members must be on a dimension tagged as Accounts.

Syntax

```
CALC TWOPASS;
```

Notes

Member formulas are applied at each consolidated level of the database. All non two-pass members are ignored during this process.

Example

```
CALC TWOPASS;
```

CCONV

Calculates currency conversions. This command is available only if your company has purchased the Currency Conversion option.

Syntax

```
CCONV currExchMbr | TOLOCALRATE curType;
```

Parameter	Description
<i>currExchMbr</i>	Currency name containing the required exchange rate. This is a member from the currency database.
TOLOCALRATE	Converts a converted currency back to the original, local rate.
<i>curType</i>	Currency type. This is a member from the CurType dimension in the currency database.

Notes

You convert data values from a local to a common, converted currency using the CCONV *currExchMbr* command. For example, you might convert data from a European currency into US\$. You can then convert the data values back to the local currency using the CCONV TOLOCALRATE *curType* command.

Note: The CCTRACK setting in the *essbase.cfg* file must be set to TRUE (the default) to enable the CCONV TOLOCALRATE command.

You can convert all or part of the main database using the rates defined in the currency database. You can keep both the local and converted values in the main database, or you can overwrite the local values with the converted values.

If you want to overwrite local values with converted values:

You do not need to create a CURPARTITION dimension in the main database. Use the CCONV command in a calculation script to convert all the data in the database.

Note: You cannot use the FIX command if the CCTRACK setting is set to TRUE (the default) in the `essbase.cfg` file and you are not using a CURPARTITION dimension.

If you want to keep both local and converted values:

In the main database, define the members that store the local and converted values. You do this by creating a CURPARTITION dimension. The CURPARTITION dimension has two partitions, one for local values and one for converted values.

► To convert data:

- 1 Use the DATACOPY command to copy data from the local to the converted partition.
- 2 Use the FIX command to calculate only the converted partition and use the CCONV command to convert the data.
- 3 Use the CALC command to recalculate the database.

To convert currencies, you must create a currency database and define specific dimensions in the main database. For more information, see the *Oracle Essbase Database Administrator's Guide*.

Example

```
CCONV YEN;
```

converts the data values from local currency values to Japanese Yen using the YEN exchange rate from the currency database.

```
CCONV TOLOCALRATE "Act xchg";
```

converts the data values back to the local currencies using the Act xchg currency type from the currency database.

```
CCONV Actual->US$;
```

converts the data values from local currencies to US\$ using the Actual, US\$ exchange rate from the currency database.

```
FIX (Act)
  CCONV TOLOCALRATE "Act xchg";
ENDFIX
```

converts the data in the Act currency partition back to the local currencies using the Act xchg currency type from the currency database.

```
DATACOPY Act TO Actual;
FIX (Actual)
  CCONV "Act xchg"->US$;
ENDFIX
CALC ALL;
```

copies Actual data values from the local currency partition to the converted currency partition. Fixes on the Actual data (in the converted partition) and converts it using the Act xchg, US\$ rate from the currency database. Recalculates the database.

See Also

- [SET UPTOLOCAL](#)
- [SET CCTRACKCALC](#)
- [CLEARCCTRACK](#)
- [“CCTRACK” on page 408](#)

CLEARBLOCK

Sets cell values to #MISSING, and if all the cells are empty or #MISSING, removes the block. This command is useful when you need to clear old data values across blocks before loading new values.

CLEARBLOCK helps optimize database calculation speed. For example, if an initial calculation creates numerous consolidated level blocks, subsequent recalculations take longer, because Essbase must pass through the additional blocks. CLEARBLOCK clears blocks before a calculation occurs.

Another example: if a database to be copied contains a lot of empty blocks, copying the database also copies the empty blocks, resulting in a many more empty blocks. Using CLEARBBLK EMPTY first makes the copy process more efficient.

If you use CLEARBLOCK within a FIX command, Essbase clears only the cells within the fixed range, and not the entire block.

Syntax

```
CLEARBLOCK ALL | UPPER | NONINPUT | DYNAMIC | EMPTY;
```

Parameter	Description
ALL	Clears and removes all blocks.
UPPER	Clears consolidated level blocks.
NONINPUT	Clears blocks containing derived values. Applies to blocks that are completely created by a calculation operation. Cannot be a block into which any values were loaded.
DYNAMIC	Clears blocks containing values derived from Dynamic Calc and Store member combinations.
EMPTY	Removes empty blocks (blocks where all values are #MISSING).

Notes

- If you regularly enter data values directly into a consolidated level, the UPPER option overwrites your data. In this case, you should use the NONINPUT option, which only clears blocks containing calculated values.

- If you use CLEARBLOCK EMPTY, the resulting, smaller database can be processed more efficiently; however, the CLEARBLOCK EMPTY process itself can take some time, depending on the size and density of the database.
- If CLEARBLOCK is used within a FIX command on a dense dimension, the FIX statement is ignored and all blocks are scanned for missing cells.

Example

```
CLEARBLOCK ALL;
CLEARBLOCK UPPER;
CLEARBLOCK NONINPUT;
CLEARBLOCK DYNAMIC;
CLEARBLOCK EMPTY;
```

See Also

- [CLEARDATA](#)

CLEARCCTRACK

Clears the internal exchange rate tables created by the “CCTRACK” on page 408 setting.

Syntax

```
CLEARCCTRACK;
```

Notes

Use this command after a data load, to reset the exchange rate tables before rerunning a currency conversion. You can use this command inside a FIX statement to clear the exchange rates for a currency partition.

Example

```
CLEARDATA Actual;
FIX(Actual)
CLEARCCTRACK;
ENDFIX
```

Clears the Actual data, fixes on the Actual data (in the converted partition) and clears the internal exchange rate tables for the Actual data.

See Also

- [“CCTRACK” on page 408](#)
- [SET CCTRACKCALC](#)
- [CCONV](#)
- [SET UPTOLOCAL](#)

CLEARDATA

Clears data values from the database and sets them to #MISSING.

This command is useful when you need to clear existing data values before loading new values into a database. CLEARDATA can only clear a section of a database. It cannot clear the entire database. To clear the entire database:

- Use the following MaxL statement:

```
alter database <db-name> reset; or
```
- Use Administration Services. *Oracle Essbase Administration Services Online Help*

CLEARDATA does not clear blocks, even if all data values in a block are #MISSING. Use CLEARBLOCK if you wish to clear blocks from the database, which can improve performance.

Syntax

```
CLEARDATA mbrName;
```

Parameter	Description
-----------	-------------

<i>mbrName</i>	Any valid single member name or member combination, or a function that returns a single member or member combination.
----------------	-----------------------------------------------------------------------------------------------------------------------

Notes

CLEARDATA does not work if placed in an IF statement.

Example

```
CLEARDATA Budget;
```

clears all Budget data.

```
CLEARDATA Budget->Colas;
```

clears only Budget data for the Colas product family.

See Also

- [CLEARBLOCK](#)

DATACOPY

Copies a range of data cells to another range within the database.

This command is useful when you must maintain an original set of data values and perform changes on the copied data set.

DATACOPY is commonly used as part of the Currency Conversion process. DATACOPY is also useful when you need to define multiple iterations of plan data.

To reduce typing, if any dimension(s) represented by the members in *mbrName1* are not represented in *mbrName2*, then by default the same member or members from *mbrName1* are assumed to exist in *mbrName2* to complete the range. The reverse is not true. Any dimension explicitly represented in *mbrName2* MUST be represented by another member of the same dimension in *mbrName1*.

The ranges specified by both *mbrName1* and *mbrName2* must be of the same size. The same dimensions represented by the members that make up *mbrName1* must also be present in *mbrName2*.

Syntax

```
DATACOPY mbrName1 TO mbrName2;
```

Parameter	Description
-----------	-------------

<i>mbrName1</i> and <i>mbrName2</i>	Any valid single member name or member combination.
-------------------------------------	-----------------------------------------------------

Notes

- The size of the copied dimensions must be equal to the destination (TO) size.
- DATACOPY follows the rules for any defined FIX command.
- To prevent creation of #MISSING blocks, add the following calculation command to your script:

```
SET COPYMISSINGBLOCK OFF;
```

Example

```
DATACOPY Plan TO Revised_Plan;
```

See Also

- [SET COPYMISSINGBLOCK](#)

DATAEXPORT

Writes data to a text file, binary file, or as direct input to a relational file using ODBC.

Syntax

For a text output file:

```
DATAEXPORT "File" "delimiter" "fileName" "missingChar"
```

For a binary output file:

```
DATAEXPORT "Binfile" "fileName"
```

For direct export to a relational database using ODBC:

```
DATAEXPORT "DSN" "dsnName" "tableName" "userName" "password"
```

Parameter	Description
-----------	-------------

"File""Binfile""DSN"	Required keyword for the type of output file. Specify the appropriate keyword, then use the associated syntax.
----------------------	----------------------------------------------------------------------------------------------------------------

"delimiter"	Required for "File" exports The character that separates fields; for example, "," Do not use with "Binfile" or "DSN" exports
-------------	------------------------------------------------------------------------------------------------------------------------------------

Parameter	Description
"fileName"	<p>Required for "File" and "Binfile" exports</p> <p>Full path name for the export file.</p> <p>Do not use with "DSN" exports.</p>
"missingChar"	<p>Optional for output type "File"</p> <ul style="list-style-type: none"> • A text string to represent missing data values. Maximum length: 128 characters. • "NULL" to skip the field, resulting in consecutive delimiters (such as ,,). • Default value: #MI <p>Do not use with "Binfile" or "DSN" exports, or in combination with the SET DATAEXPORTRELATIONALFILE command.</p>
"dsnName"	<p>Required for output type "DSN"</p> <p>The DSN name used to communicate with the SQL database. A substitution variable can be used.</p> <p>Do not use with output type "File" or "Binfile."</p>
"tableName"	<p>Required for "DSN" exports</p> <p>Name of the table where the exported data is to be inserted. The table must exist, and table and column names cannot contain spaces.</p> <p>Do not use with "File" or "Binfile" exports.</p>
"userName"	<p>Required for "DSN" exports</p> <p>The user name that is used when communicating with the database. A substitution variable can be used.</p> <p>Do not use with "File" or "Binfile" exports.</p>
"password"	<p>Required for "DSN" exports</p> <p>The password that is used when communicating with the database. A substitution variable can be used.</p> <p>Do not use with "File" or "Binfile" exports.</p>

Notes

- In general, specify SET commands within the calculation script to specify various options, and then use FIX...ENDFIX to refine data to be exported, including the DATAEXPORT command within the FIX...ENDFIX command set. Without FIX...ENDFIX, the entire database is exported.
- If outputting a file, and *fileName*:
 - Does not include a path, the file is written in the application directory.
 - Includes a path, Essbase interprets the path in context to the server. Export files cannot be written to a client.
- When using DATAEXPORT "DSN" to export data for direct insertion to a relational database:
 - You can use the [“DATAEXPORTENABLEBATCHINSERT” on page 414](#) configuration setting to enable the batch insert method, which is faster than the default row-insert

method. With batch insert, Essbase determines the batch size, but you can use the “[DEXPSQLROWSIZE](#)” on [page 418](#) configuration setting to specify the number of rows (from 2 to 1000) to be batch inserted. Essbase inserts the rows when the specified batch size is reached.

- The table to which the data is to be written must exist prior to data export
- Table and column names cannot contain spaces.

Note: 64-bit Essbase does not support using the DATAEXPORT batch-insert method to export data directly into a SQL data source.

- The export process does not begin if users are logged into the database. After the export process begins, the database is in read-only mode. Users can read the data but they cannot change it. After the export process is finished, Essbase returns the database to read-write mode and users can make changes to the data.
- Use the [DATAIMPORTBIN](#) command to import a previously exported binary export file.

Description

The DATAEXPORT calculation command writes data into a text or binary output file, or connects directly to an existing relational database wherein the selected exported data is inserted.

Whereas both the MaxL [Export Data](#) statement and the ESSCMD [EXPORT](#) command can export all, level 0, or input data from the entire database as text data, the DATAEXPORT calculation command also enables you to:

- Use [FIX...ENDFIX](#) or [EXCLUDE...ENDEXCLUDE](#) calculations to select a slice of the database and use a [DATAEXPORTCOND](#) command to select data based on data values.
- Use parameters to qualify the type and destination of the export data.
- Use options provided by the [SET DATAEXPORTOPTIONS](#) command to refine export content, format, or process.
- Use the [SET DATAIMPORTIGNORETIMESTAMP](#) command to manage the import requirement for a matching outline timestamp.

Using Report Writer to create an "export" file also provides extensive flexibility in selecting and formatting the data; however, using DATAEXPORT outputs the data more quickly. For information about using Report Writer to export data, see the *Oracle Essbase Database Administrator's Guide*.

Example

Text Output File Example 1

```
SET DATAEXPORTOPTIONS
{
  DataExportLevel "LEVEL0";
};
DATAEXPORTCOND ("Sales">=1000);
FIX ("100-10", "New York", "Actual", "Sales");
```

```
DATAEXPORT "File" ", " "b:\exports\jan.txt" "#MI";
ENDFIX;
```

Specifies a level 0 data export level, limits output to data only with 1000 or greater Sales, fixes the data slice, then exports to a text file located at b:\exports\jan.txt, using comma (,) delimiters and specifying #MI for missing data values.

Text Output File Example 2

```
SET DATAEXPORTOPTIONS
{
  DataExportLevel "LEVEL0";
  DataExportRelationalFile ON;
};
DATAEXPORTCOND ("Sales">=1000);
FIX ("100-10", "New York", "Actual", "Sales");
DATAEXPORT "File" ", " "b:\exports\jan.txt";
ENDFIX;
```

Specifies the same export content as Example 1; however, the output file is formatted for input to a relational database. Notice the *missingChar* parameter is intentionally excluded.

Binary Example 1: Export

```
SET DATAEXPORTOPTIONS
{
  DataExportLevel "ALL";
};
FIX ("New York");
DATAEXPORT "BinFile" "b:\backup\newyork.bin";
ENDFIX;
```

Exports all New York blocks. Binary exports can be fixed only on sparse dimensions. Essbase uses the same bitmap compression technique to create the file as is used by Essbase Kernel.

Binary Example 2: Import

```
SET DATAIMPORTIGNORETIMESTAMP OFF;
DATAIMPORTBIN "b:\backup\newyork.bin"
```

Imports the previously exported file. The timestamp must match. The data is imported to the database on which the calculation script is executed. Because only data was exported, to recreate a database after using DATAIMPORT to read in the data, you must recalculate the data.

Direct Input to Relational Database Example

```
SET DATAEXPORTOPTIONS
{
  DataExportLevel "ALL";
};
FIX("100-10", "New York", "Actual", "Sales");
DATAEXPORT "DSN" "cur_sale" "newyork" "admin" "password";
ENDFIX;
```

Inserts the selected records directly to the table named newyork. By default, Essbase inserts exported data row-by-row. If the [“DATAEXPORTENABLEBATCHINSERT”](#) on page 414

configuration setting is set to TRUE in `essbase.cfg`, records are batch inserted. To control the number of rows that are batch inserted at a time, use the “DEXPSQLROWSIZE” on page 418 configuration setting in conjunction with “DATAEXPORTENABLEBATCHINSERT” on page 414 set to TRUE.

See Also

- [FIX...ENDFIX](#)
- [SET Commands](#)
- [“DATAEXPORTENABLEBATCHINSERT” on page 414](#)
- [“DEXPSQLROWSIZE” on page 418](#)
- [SET DATAEXPORTOPTIONS](#)
- [SET DATAIMPORTIGNORETIMESTAMP](#)
- [DATAEXPORTCOND](#)
- [DATAIMPORTBIN](#)

DATAEXPORTCOND

Specifies value conditions that select export records to be included or marked as “#NoValue” in the export output file.

Syntax

```
DATAEXPORTCOND "conditionExpression" ReplaceAll;
```

Parameter	Description
conditionExpression	<p>One or more conditions separated by a logical AND or OR. Each condition specifies a member name the value of which is equal to (=), greater than (>), greater than or equal (>=), less than (<), or less than or equal (<=) to a specified value or the value of another member; for example, "Sales" > 500 AND "Ending Inventory" < 0.</p> <p>The condition list is processed from left to right. Thus the result of cond1 is calculated first, then the operator (AND or OR) is calculated against cond2, and so on. While processing conditions, if a resultant condition is found to be false, the entire record is omitted from the output file</p>
ReplaceAll	<p>The keyword that indicates whether exported records are to be excluded from the initial export set of records, or included but marked as “#NoValue”. The initial export set of records is determined by the region defined by the FIX command and SET commands that apply to the data export.</p> <ul style="list-style-type: none">● When ReplaceAll is not specified, only those records within the initial export set are exported that meet the specified conditions.● When ReplaceAll is specified, all records within the initial export set are exported, but the AND and OR specifications are ignored. All fields that do not satisfy any of the specified conditions are marked as #NoValue.

Notes

Use DATAEXPORTCOND to specify conditions that identify records to be exported based on field values. Whether a condition can specify a member compared to a numeric value or compared to another member depends the member being a row or column element of the output. In order to represent multidimensional data within a two-dimension file, the members of one dense dimension become columns. The combinations of the members of the other dense

dimensions and the sparse dimensions create rows. (You can use the DataExportColHeader option of the [SET DATAEXPORTOPTIONS](#) calculation command to specify which dimension defines the columns.)

- If a condition is placed on a "column" member, the value of the specified member can be compared to a specific value (for example, Sales > 500) or to the value of another member of the same export record (for example, Sales < Cost).
- If a condition is placed on a "row" member, the value of the specified member can be compared only to a specific value (for example, Cost < 500).

Example

Not Using ReplaceAll

```
SET DATAEXPORTOPTIONS
{
  DataExportLevel "ALL";
};
DATAEXPORTCOND (Actual >= 2 AND Sales > 2000 OR COGS > 600);
FIX("100-10", "East");
  DATAEXPORT "File" " ", "E:\temp\2222.txt";
ENDFIX;
```

Sets the contents of the initial export file through the DataExportLevel option of the [SET DATAEXPORTOPTIONS](#) command and [FIX...ENDFIX](#) command. The DATAEXPORTCOND command specifies the records to be included when the Actual value is greater than or equal to 2 and Sales are greater than 2000, or when the Actual value is greater than or equal to 2 and COGS is greater than 600. The conditions are specified on the column Actual, the column Sales, and the column COGS. The exported data includes only records that meet the conditions.

Sample output:

```
"Sales", "COGS", "Marketing", "Payroll", "Misc", "Opening Inventory", "Additions", "Ending
Inventory"
"100-10", "East"
"Jun", "Actual", 2205, 675, 227, 177, 2, 3775, 2028, 3598
"Jul", "Actual", 2248, 684, 231, 175, 2, 3598, 1643, 2993
"Sep", "Actual", 2012, 633, 212, 175, 4, 2389, 1521, 1898
"Jun", "Budget", 2070, 620, 180, 120, #Mi, 2790, 1700, 2420
"Jul", "Budget", 2120, 620, 180, 120, #Mi, 2420, 1400, 1700
"Aug", "Budget", 2120, 620, 180, 120, #Mi, 1700, 1400, 980
```

Using ReplaceAll

```
SET DATAEXPORTOPTIONS
{
  DataExportLevel "ALL";
};
DATAEXPORTCOND (Actual >= 2 AND Sales > 2000 OR COGS > 600);
FIX("100-10", "East");
  DATAEXPORT "File" " ", "E:\temp\2222.txt" ReplaceAll;
ENDFIX;
```

Using the same conditions as the prior example, but including "ReplaceAll" in the DATAEXPORT command, the exported data includes all records specified by the FIX command. #NoValue is inserted for fields that do not meet the specified conditions. Sample output:

```

"Sales", "COGS", "Marketing", "Payroll", "Misc", "Opening Inventory", "Additions", "Ending
Inventory" "100-10", "East" "Jan", "Actual", #NoValue, #NoValue, 199, 175, 2, 4643, 1422, 4253
"Feb", "Actual", #NoValue, #NoValue, 196, 175, 3, 4253, 1413, 3912
"Mar", "Actual", #NoValue, #NoValue, 199, 175, 3, 3912, 1640, 3747
"Apr", "Actual", #NoValue, 606, 204, 177, 3, 3747, 1824, 3701
"May", "Actual", #NoValue, 622, 210, 177, 4, 3701, 2023, 3775
"Jun", "Actual", 2205, 675, 227, 177, 2, 3775, 2028, 3598
"Jul", "Actual", 2248, 684, 231, 175, 2, 3598, 1643, 2993
"Aug", "Actual", 2245, 684, 231, 175, #NoValue, 2993, 1641, 2389
"Sep", "Actual", 2012, 633, 212, 175, 4, 2389, 1521, 1898
"Oct", "Actual", #NoValue, #NoValue, 196, 175, 3, 1898, 1535, 1677
"Nov", "Actual", #NoValue, #NoValue, 192, 175, #NoValue, 1677, 1584, 1553
"Dec", "Actual", #NoValue, #NoValue, 200, 175, 2, 1553, 1438, 1150
"Jan", "Budget", #NoValue, #NoValue, 160, 120, #Mi, 4490, 1100, 3900
"Feb", "Budget", #NoValue, #NoValue, 160, 120, #Mi, 3900, 1200, 3460
"Mar", "Budget", #NoValue, #NoValue, 160, 120, #Mi, 3460, 1400, 3170
"Apr", "Budget", #NoValue, #NoValue, 150, 120, #Mi, 3170, 1500, 2920
"May", "Budget", #NoValue, #NoValue, 160, 120, #Mi, 2920, 1700, 2790
"Jun", "Budget", 2070, 620, 180, 120, #Mi, 2790, 1700, 2420
"Jul", "Budget", 2120, 620, 180, 120, #Mi, 2420, 1400, 1700
"Aug", "Budget", 2120, 620, 180, 120, #Mi, 1700, 1400, 980
"Sep", "Budget", #NoValue, #NoValue, 150, 120, #Mi, 980, 1300, 390
"Oct", "Budget", #NoValue, #NoValue, 110, 70, #Mi, 390, 1180, 110
"Nov", "Budget", #NoValue, #NoValue, 150, 120, #Mi, 110, 1460, 60
"Dec", "Budget", #NoValue, #NoValue, 150, 120, #Mi, 60, 1300, -260

```

See Also

- [DATAEXPORT](#)
- [FIX...ENDFIX](#)
- [SET Commands](#)
- [SET DATAEXPORTOPTIONS](#)

DATAIMPORTBIN

Imports the binary output file previously exported with the [DATAEXPORT](#) "Binfile" calculation command.

You can use DATAIMPORTBIN to import previously exported binary files. For example, you can use DATAEXPORT "Binfile" and DATAIMPORTBIN as a method for data backup and recovery.

Syntax

```
DATAIMPORTBIN fileName;
```

Parameter Description

`fileName` Full path name for the binary input file to be imported.

Notes

- The outline timestamp is included with the export file created by DATAEXPORT. By default, the DATAIMPORTBIN process checks the timestamp. Use the SET

DATAIMPORTIGNORETIMESTAMP calculation command with DATAIMPORT to bypass checking the timestamp. See [SET DATAIMPORTIGNORETIMESTAMP](#) for details.

- Use DATAIMPORTBIN only with files created by DATAEXPORT "Binfile".

Example

```
DATAIMPORTBIN e:\january\sales.bin;
```

Specifies the binary file e:\january\sales.bin is to be imported to the database for which the calculation script is being run.

See Also

- [DATAEXPORT](#)
- [SET DATAIMPORTIGNORETIMESTAMP](#)

ELSE

The ELSE command designates a conditional action to be performed in an IF statement. All actions placed after the ELSE in an IF statement are performed only if the test in the IF statement generates a value of FALSE.

Syntax

```
ELSE statement ; [ ...statement; ] ENDIF;
```

Parameter Description

statement Those operations that are to be performed in the event that the IF test including the ELSE command produces a FALSE, or 0, result.

Notes

- The ELSE command can only be used in conjunction with an IF command.
- You do not need to end ELSE statements with ENDIF statements. Only IF statements should be ended with ENDIF statements.

Example

The following example is based on the Sample Basic database. This calculation script tests to see if the current member in the Market dimension is a descendant of West or East. If so, Essbase multiplies the value for Marketing by 1.5. If the current member is not a descendant of West or East, Essbase multiplies the value for Marketing by 1.1.

```
Marketing
(IF (@ISMBR(@DESCENDANTS(West))
OR
(@ISMBR(@DESCENDANTS(East))))
Marketing = Marketing * 1.5;
ELSE
Marketing = Marketing * 1.1;
ENDIF;
```

See Also

- [ELSEIF](#)
- [ENDIF](#)
- [IF](#)

ELSEIF

Designates a conditional test and conditions that are performed if the preceding IF test generates a value of FALSE. For this reason, multiple ELSEIF commands are allowed following a single IF.

Syntax

```
ELSEIF( condition ) statement ; [ ...statement ; ]  
ELSEIF | ELSE | ENDIF
```

Parameter Description

condition Formula or function that returns a Boolean value of TRUE (a nonzero value) or FALSE (a zero value).

statement Those operations that are to be performed in the event that the IF test (including the ELSE command) produces a FALSE, or 0, result.

Notes

- The ELSEIF command must be used in conjunction with an IF command.
- You do not need to end ELSEIF statements with ENDIF statements. Only IF statements should be ended with ENDIF statements. For example:

```
IF (condition)  
    statement;  
IF (condition)  
    statement;  
ELSEIF (condition)  
    statement;  
ENDIF;  
statement;  
ENDIF;
```

Example

The following example is based on the Sample Basic database. This calculation script tests to see if the current member in the Market dimension is a descendant of West or East. If so, Essbase multiplies the value for Marketing by 1.5. The calculation script then tests to see if the current member is a descendant of South. If so, Essbase multiplies the value for Marketing by .9. If the current member is not a descendant of West, East, or South, Essbase multiplies the value for Marketing by 1.1.

```
IF (@ISMBR(@DESCENDANTS(West))  
    OR  
    @ISMBR(@DESCENDANTS(East))  
    )  
    Marketing = Marketing * 1.5;  
ELSEIF (@ISMBR(@DESCENDANTS(South))  
    )
```

```
Marketing = Marketing * .9;
ELSE
Marketing = Marketing * 1.1;
ENDIF;
```

See Also

- [ELSE](#)
- [ENDIF](#)
- [IF](#)

ENDIF

Marks the end of an IF command sequence. The ENDIF command can be used only in conjunction with IF or IF ... ELSEIF statements.

Syntax

```
ENDIF;
```

Notes

- You must supply an ENDIF statement for every IF statement in your formula or calculation script. If you do not supply the required ENDIF statements, your formula or calculation script does not verify.
- If you are using an IF statement nested within another IF statement, end each IF with an ENDIF. For example:

```
"Opening Inventory"
  (IF (@ISMBR(Budget))
    IF (@ISMBR(Jan))
      "Opening Inventory" = Jan;
    ELSE
      "Opening Inventory" = @PRIOR("Ending Inventory");
    ENDIF;
  ENDIF;)
```

- You do not need to end ELSE or ELSEIF statements with ENDIF statements.
- Although ending ENDIF statements with a semicolon is not required, it is good practice to follow each ENDIF statement in your formula or calculation script with a semicolon.
- IF, ELSE, ELSEIF, and ENDIF must all be used within a database outline formula, or must be associated with a member in the database outline when used in a calculation script. For more information, see the *Oracle Essbase Database Administrator's Guide*.

Example

The following example is based on the Sample Basic database. This calculation script tests to see if the current member in the Market dimension is a descendant of West or East. If so, Essbase multiplies the value for Marketing by 1.5. The calculation script then tests to see if the current member is a descendant of South. If so, Essbase multiplies the value for Marketing by .9. If the current member is not a descendant of West, East, or South, Essbase multiplies the value for Marketing by 1.1.


```

IF (@ISMBR(@DESCENDANTS(West))
    OR
    @ISMBR(@DESCENDANTS(East)))
    Marketing = Marketing * 1.5;
ELSEIF(@ISMBR(@DESCENDANTS(South)))
    Marketing = Marketing * .9;
ELSE
    Marketing = Marketing * 1.1;
ENDIF;

```

See Also

- [ELSE](#)
- [ELSEIF](#)
- [IF](#)

EXCLUDE...ENDEXCLUDE

The EXCLUDE command allows you to define a fixed range of members which are not affected by the associated commands. The ENDEXCLUDE command ends an EXCLUDE command block.

As shown in the example, you call ENDEXCLUDE after all of the commands in the EXCLUDE command block have been called, and before the next element of the calculation script.

Specifying members that should not be calculated in an EXCLUDE..ENDEXCLUDE command may be simpler than specifying a complex combination of member names in a FIX...ENDFIX command.

Syntax

```

EXCLUDE (Mbrs)
COMMANDS ;
ENDEXCLUDE

```

Parameter	Description
-----------	-------------

- | | |
|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Mbrs | A member name or list of members from any number of database dimensions. <i>Mbrs</i> can also contain: <ul style="list-style-type: none"> ● AND/OR operators. Use the AND operator when all conditions must be met. Use the OR operator when one condition of several must be met. ● Member set functions, which are used to build member lists based on other members. |
|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

COMMANDS The commands to be executed for the duration of the EXCLUDE.

Notes

- Use EXCLUDE...ENDEXCLUDE commands only within calculation scripts, not in outline member formulas.
- You can include EXCLUDE commands within FIX command blocks.

- If a FIX command within an EXCLUDE command block specifies cells already specified by the EXCLUDE statement, those cells are not calculated, and a warning message is posted to the application log file.
- An EXCLUDE command block cannot include CALC ALL, CLEARDATA, and DATACOPY commands.
- AND and OR operators have the same precedence and are evaluated from left to right. Use parentheses to group the expressions. For example: A OR B AND C is the same as (A OR B) AND C). However, subexpressions (for example, (A OR (B AND C)) are evaluated before the whole expression, producing a different result.
- Inside EXCLUDE command blocks, the AND operator represents the intersection of two sets; the OR operator represents the union of two sets. In formulas, these operators are Boolean operators. Using the AND or OR operators on members that are from different dimensions, returns:
 - AND: An empty set. The EXCLUDE statement is ignored and the calculation continues with a warning message.
 - OR: The union of two members sets. EXCLUDE (Jan OR Market) is identical to FIX (Jan, Market).
- NOT operators are not supported in EXCLUDE command blocks. Use the @REMOVE function.
- You do not need to follow ENDEXCLUDE with a semicolon.
- Use the @ATTRIBUTE and @WITHATTR functions to specify attributes within EXCLUDE command blocks; for example EXCLUDE (@ATTRIBUTE (Can)) . FIX (Can) is not supported.
- You cannot use EXCLUDE on a dimension if it is a subset of a dimension that you calculate within the EXCLUDE command block. For example you could not use Market "New Mkt" in an EXCLUDE statement if you calculate all of Market within the command block.
- Dynamic Calc members are ignored in an EXCLUDE statement. If the only member in an EXCLUDE statement is a Dynamic Calc member, an error message is displayed stating that the EXCLUDE statement cannot contain a Dynamic Calc member.
- If the EXCLUDE command is issued from a calculation script and produces an empty set, that part of the calculation is ignored, and the calculation continues to the next statement. The application log entry for the calculation shows that the EXCLUDE statement evaluated to an empty set (Calculating [...] with fixed members []).

For example, consider the following statement in a Sample Basic calculation script:

```
EXCLUDE (@children(Jan))
CALC DIM (Accounts, Product, Market)
ENDEXCLUDE
```

Since @children(Jan) is empty (Jan is a level 0 member), the EXCLUDE parameter is ignored; the calculation operates on the entire database.

Similarly, if a region defining a partition or a security filter evaluates to an empty set, Essbase behaves as if the region definition or security filter does not exist.

- Calculator function @RANGE and the cross-dimensional operator (->) cannot be used inside an EXCLUDE *Mbrs* parameter).

Example

The following example excludes calculations on the children of Qtr4, enabling calculation of other quarters in the Year dimension.

```
EXCLUDE (@CHILDREN(Qtr4))
CALC DIM (Year)
ENDEXCLUDE
```

See Also

- [FIX...ENDFIX](#)
- [LOOP...ENDLOOP](#)

FIX...ENDFIX

The FIX...ENDFIX command block restricts database calculations to a subset of the database. All commands nested between the FIX and ENDFIX statements are restricted to the specified database subset.

This command is useful because it allows you to calculate separate portions of the database using different formulas, if necessary. It also allows you to calculate the sub-section much faster than you would otherwise.

The ENDFIX command ends a FIX command block. As shown in the example, you call ENDFIX after all of the commands in the FIX command block have been called, and before the next element of the calculation script.

Syntax

```
FIX (fixMbrs)
COMMANDS ;
ENDFIX
```

Parameter	Description
-----------	-------------

- | | |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| fixMbrs | A member name or list of members from any number of database dimensions. <i>fixMbrs</i> can also contain: <ul style="list-style-type: none"> • AND/OR operators. Use the AND operator when all conditions must be met. Use the OR operator when one condition of several must be met. • Member set functions, which are used to build member lists based on other members. |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

COMMANDS The commands you want to be executed for the duration of the FIX.

Notes

- You can use [SET EMPTYMEMBERSETS](#) to stop the calculation within a FIX command if the FIX evaluates to an empty member set.
- FIX commands can be nested within other FIX command blocks. For an example of an incorrect use of nested FIX commands, see “Using the FIX Command” in the *Oracle Essbase Database Administrator's Guide*.

- FIX statements can only be used in calculation scripts, not in outline member formulas. Use an IF command instead of a FIX statement in member formulas. For example:

```
Jan(
  IF (Sales)
  Actual=5;
  ENDFIX;)
```

- AND/OR operators have the same precedence; Essbase evaluates them from left to right. Use parentheses to group the expressions. For example: A OR B AND C is the same as ((A OR B) AND C). However, if you use (A OR (B AND C)), Essbase evaluates the sub-expression in parentheses (B AND C) before the whole expression, producing a different result.
- Inside FIX statements, the AND operator represents the intersection of two sets; the OR operator represents the union of two sets. In formulas, these operators are Boolean operators. Using the AND or OR operators on members that are from different dimensions, returns:
 - AND: An empty set. The FIX statement is ignored and the calculation continues with a warning message.
 - OR: The union of two members sets. FIX (Jan OR Market) is identical to FIX (Jan, Market).
- In FIX statements, members from the same dimension are always acted on as OR unless you specify otherwise.
- NOT operators are not supported in FIX statements. Use the @REMOVE function with FIX statements.
- You do not need to follow ENDFIX with a semicolon.
- You can specify attributes in FIX statements using the @ATTRIBUTE and @WITHATTR functions; for example FIX (@ATTRIBUTE (Can)). You must use these functions; FIX (Can) is not supported.
- You cannot use a FIX statement on a dimension if it is a subset of a dimension that you calculate within the FIX statement. For example you could not use Market "New Mkt" in a FIX statement if you calculate all of Market within the FIX statement.
- Dynamic Calc members are ignored in a FIX statement. If the only member in a FIX statement is a Dynamic Calc member, an error message is displayed stating that the FIX statement cannot contain a Dynamic Calc member.
- If the FIX command is issued from a calculation script and produces an empty set, that part of the calculation is ignored, and the calculation continues to the next statement. The application log entry for the calculation shows that the FIX statement evaluated to an empty set (Calculating [...] with fixed members []).

For example, using Sample Basic, assume this statement is in a calculation script:

```
FIX (@children(Jan))
  CALC DIM (Accounts, Product, Market)
  ENDFIX
```

Since @children(Jan) is empty, the FIX is ignored; the calculation issues a warning and operates on the entire database.

Similarly, if a region defining a partition or a security filter evaluates to an empty set, Essbase issues a warning and behaves as if the region definition or security filter did not exist.

- The calculator function @RANGE and the cross-dimensional operator (->) cannot be used inside a FIX *fixMbrs* parameter.
- Using an EXCLUDE...ENDEXCLUDE command to specifying members that should not be calculated may be simpler than specifying a complex combination of member names in a FIX...ENDFIX command.

Example

```
FIX (Budget)
  CALC DIM (Year, Measures, Product, Market);
ENDFIX
FIX (Budget, Jan, Feb, Mar, @DESCENDANTS(Profit))
  CALC DIM (Product, Market);
ENDFIX
```

The following example fixes on the children of East and the Market dimension members with the UDA "New Mkt".

```
FIX (@CHILDREN(East) OR @UDA(Market, "New Mkt"))
```

The following example fixes on the children of East with the UDA "New Mkt" and Market dimension members with the UDA "Big Mkt".

```
FIX((@CHILDREN(East) AND @UDA(Market, "New Mkt")) OR @UDA(Market, "Big Mkt"))
```

See Also

- [EXCLUDE...ENDEXCLUDE](#)
- [LOOP...ENDLOOP](#)
- [SET EMPTYMEMBERSETS](#)

IF

Performs conditional tests within a formula. Using the IF statement, you can define a Boolean test, as well as formulas to be calculated if the test returns either a TRUE or FALSE value.

Syntax

```
IF( condition ) statement ; [ ...statement ; ] [ ELSEIF...statement | ELSE...statement ]
  ENDF;
```

Parameter Description

condition Formula or function that returns a Boolean value of TRUE (a nonzero value) or FALSE (a zero value).

statement Operations to be performed depending on the results of the test.

Notes

- The IF statement block can also use the ELSE and ELSEIF statements as part of its decision syntax.
- For information about using ENDIF statements and semicolons with IF, ELSE, and ELSEIF statements, see [ENDIF](#).
- In calculation scripts, IF statements must be placed within parentheses and associated with a specific database member. They must also be closed with ENDIF statements. For more information, see the *Oracle Essbase Database Administrator's Guide*.
- You can specify attributes in IF statements using the @ATTRIBUTE and @WITHATTR functions; for example IF (@ISMBR(@ATTRIBUTE(Can))) . . . You must use these functions; IF (@ISMBR(Can)) is not supported.

Example

Example 1

```
IF (
    @ISMBR(@DESCENDANTS(Europe))
OR   @ISMBR(@DESCENDANTS(Asia))
)
    Taxes = "Gross Margin" * "Foreign Tax Rate";
ELSE
    Taxes = "Gross Margin" * "Domestic Tax Rate";
ENDIF;
```

This test checks to see if the current cell includes a member that is a descendant of either the Europe or Asia members. If it does, the formula calculates the taxes for the member based on the foreign tax rate. If the current cell does not include a member from one of those groups, then the domestic tax rate is used for the tax calculation.

Example 2

When you use an IF statement as part of a member formula in a calculation script, you need to perform both of the following tasks:

- Associate the IF statement with a single member
- Enclose the IF statement in parentheses

A sample IF statement is illustrated in the following example:

```
Profit
(IF (Sales > 100)
    Profit = (Sales - COGS) * 2;
ELSE
    Profit = (Sales - COGS) * 1.5;
ENDIF;)
```

Essbase cycles through the database and performs the following calculations:

1. The IF statement checks to see if the value of Sales for the current member combination is greater than 100.

2. If Sales is greater than 100, Essbase subtracts the value in COGS from the value in Sales, multiplies the difference by 2, and places the result in Profit.
3. If Sales is less than or equal to 100, Essbase subtracts the value in COGS from the value in Sales, multiplies the difference by 1.5, and places the result in Profit.

The whole of the IF ... ENDIF statement is enclosed in parentheses and associated with the Profit member, Profit (IF(...)).

See Also

- [ELSE](#)
- [ELSEIF](#)
- [ENDIF](#)

LOOP...ENDLOOP

The LOOP...ENDLOOP command block specifies the number of times to iterate calculations. All commands between the LOOP and ENDLOOP statements are performed the number of times that you specify.

Syntax

```
LOOP (integer, [break]) COMMANDS ;
ENDLOOP
```

Parameter	Description
integer	The integer constant that indicates the number of times to execute the commands contained in the loop block.
break	Optional parameter used to break the iterative process of a loop. <i>break</i> must be the name of a temporary variable (VAR). Setting the value of the variable to 1 during the execution of the loop causes the loop to break at the beginning of its next iteration.

COMMANDS Those commands that you want to be executed for the duration of the LOOP.

Notes

LOOP is a block command that defines a block of commands for repeated execution. As with the FIX command, you can nest LOOP statements if necessary.

The ENDLOOP command ends a LOOP command block. It terminates the LOOP block and occurs after the commands in the LOOP block, but before any other commands.

Example

In this example, the LOOP command finds a solution for Profit and Commission. This operation is done as a loop because Profit and Commission are interdependent: Profit is needed to evaluate Commission, and Commission is needed to calculate Profit. This example thus provides a model for solving simultaneous formulas.

```
FIX("New York", Camera, Actual, Mar)
  LOOP(30)
    Commission = Profit * .15;
```

```
Profit = Margin - "Total Expenses" - Commission;
ENDLOOP;
ENDFIX
```

See Also

- [FIX...ENDFIX](#)

SET Commands

SET commands in a calculation script are procedural. The first occurrence of a SET command in a calculation script stays in effect until the next occurrence of the same SET command.

Example

In the following example, Essbase displays messages at the DETAIL level when calculating the Year dimension. However, when calculating the Measures dimension, Essbase displays messages at the SUMMARY level.

```
SET MSG DETAIL;CALC DIM(Year) ;
SET MSG SUMMARY;CALC DIM(Measures) ;
```

In the following example, Essbase calculates member combinations for Qtr1 with the [SET AGGMISSG](#) setting turned on. Essbase then does a second calculation pass through the database and calculates member combinations for East with the AGGMISSG setting turned off. For more information on calculation passes, see the *Oracle Essbase Database Administrator's Guide*.

```
SET AGGMISSG ON;Qtr1;
SET AGGMISSG OFF;East;
```

SET AGGMISSG

Specifies whether Essbase consolidates #MISSING values in the database.

The default behavior of SET AGGMISSG is determined by the global setting for the database, as described in the *Oracle Essbase Database Administrator's Guide*.

Syntax

```
SET AGGMISSG ON | OFF ;
```

Notes

SET AGGMISSG commands apply to calculating sparse dimensions.

Example

```
SET AGGMISSG OFF;
CALC ALL;
CALC PERCENTS;
```

See Also

- [SET Commands](#)

SET CACHE

Specifies the size of the calculator cache.

Syntax

```
SET CACHE HIGH | DEFAULT | LOW | OFF | ALL;
```

Parameter	Description
HIGH, DEFAULT, and LOW	Levels defining the size of the calculator cache. You set the values of HIGH, DEFAULT and LOW in the <code>essbase.cfg</code> file. If you do not set the value of DEFAULT in the <code>essbase.cfg</code> file, Essbase uses a default value of 200,000 bytes. The maximum calculator cache size that you can specify is 200,000,000 bytes.
OFF	Essbase does not use a calculator cache.
ALL	Essbase uses a calculator cache, even when you do not calculate at least one full sparse dimension.

Notes

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

You can choose one of three levels. The size of the calculator cache at each level is defined using the `CALCCACHE {HIGH | DEFAULT | LOW}` settings in the `essbase.cfg` file.

The level you choose depends on the amount of memory your system has available and the configuration of your database.

For detailed information on setting the size of your calculator cache, see the *Oracle Essbase Database Administrator's Guide*.

You can specify whether, by default, Essbase uses a calculator cache using the `CALCCACHE TRUE | FALSE` setting in the `essbase.cfg` file. By default, `CALCCACHE` is set to `TRUE`.

Essbase uses the calculator cache providing that:

- Your database has at least two sparse dimensions.
- You calculate at least one, full sparse dimension (unless you specify the `CALCCACHE ALL` option).

You can use this command more than once within a calculation script.

You can display the calculator cache setting using the `SET MSG` command.

Example

If the `essbase.cfg` file contains the following settings:

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

then:

```
SET CACHE HIGH;
```

sets a calculator cache of up to 1,000,000 bytes for the duration of the calculation script.

```
SET CACHE DEFAULT;
```

sets a calculator cache of up to 300,000 bytes for the duration of the calculation script.

```
SET CACHE LOW;
```

sets a calculator cache of up to 200,000 bytes for the duration of the calculation script.

```
SET CACHE ALL;
```

```
SET CACHE LOW;
```

sets a calculator cache of 200,000 bytes to be used even when you do not calculate at least one, full sparse dimension.

```
SET CACHE OFF;
```

means that Essbase does not use a calculator cache.

See Also

- [“CALCCACHE” on page 396](#)
- [“CALCCACHEHIGH” on page 397](#)
- [SET MSG](#)
- [SET Commands](#)

SET CALCPARALLEL

Enables parallel calculation in place of the default serial calculation.

Essbase analyzes each pass of a calculation to determine whether parallel calculation is possible. If it is not, Essbase uses serial calculation even if CALCPARALLEL is set.

Syntax

```
SET CALCPARALLEL n;
```

Parameter Description

n	A required parameter, an integer from 1-4 on 32-bit platforms or from 1-8 on 64-bit platforms, specifying the number of threads to be made available for parallel calculation. The default value specifies serial calculation: no parallel calculation takes place. Values 1-4 (1-8 on 64-bit) specifies parallel calculation with 1-4 (or 1-8) threads. Values of 0 specify serial calculation. Values less than 0 return an error. Values greater than 4 are interpreted as 4.
---	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Note: Values less than 0 treated differently than they are by the [“CALCPARALLEL” on page 406](#) configuration file setting.

Notes

- A number of features are affected by parallel calculation. See the *Oracle Essbase Database Administrator's Guide* for a list of these effects and for detailed information about how Essbase performs parallel calculation.
- If your outline generates many empty tasks, thus reducing opportunities for parallel calculation, consider setting the CALCTASKDIMS configuration setting to increase the number of tasks and to decrease the size of each task identified for parallel calculation. See the *Oracle Essbase Database Administrator's Guide* for more information about what kind of outlines or calculation scripts generate many empty tasks.
- Consider setting the value of CALCPARALLEL to one less than the number of available processors. This saves one processor for use either by the operating system or by the Essbase process that writes out dirty blocks from the calculator cache.
- You can use SET CALCPARALLEL in a calculation script to override a CALCPARALLEL entry in the configuration file.

Example

```
SET CALCPARALLEL 3;
```

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

See Also

- [SET CALCTASKDIMS](#)
- [“CALCTASKDIMS” on page 407](#)
- [“CALCPARALLEL” on page 406](#)
- [SET Commands](#)

SET CALCTASKDIMS

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

Syntax

```
SET CALCTASKDIMS n;
```

Parameter Description

- n A required parameter, an integer specifying the number of sparse dimensions to be included when Essbase identifies tasks that can be performed at the same time.
- The default value, 1, indicates that only the last sparse dimension in the outline will be 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 the 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.
- Essbase issues an error if the value is less than 1. A value greater than the number of sparse dimensions in the outline is interpreted as the largest valid value.
- Using the calculator bitmap cache can affect this value. See the *Oracle Essbase Database Administrator's Guide* discussion of parallel calculation for more information.

Note: Values less than 0 are treated differently than they are by the “CALCTASKDIMS” on page 407 configuration setting.

Notes

- A number of features are affected by parallel calculation. See the *Oracle Essbase Database Administrator's Guide* for a list of these effects and for detailed information about how Essbase performs parallel calculation.
- Use this configuration setting only if your outline generates many empty tasks, thus reducing opportunities for parallel calculation.
- If you do not notice an improvement in performance after raising the value of CALCTASKDIMS, consider returning it to its default value of 1. Sometimes using more task dimensions can generate such a large number of tasks that performance may decrease instead of increase, because the overhead of generating and managing the tasks is too great.
- You can use SET CALCTASKDIMS to override a CALCTASKDIMS entry in the configuration file. For example you might want to set all applications to use a single dimension for parallel calculation, but issue a calculation script command against a single application or database to use two dimensions.

Example

```
SET CALCTASKDIMS 2;
```

specifies that the last two sparse dimensions in the outline will be used to identify potential tasks to be performed at the same time during a calculation pass.

See Also

- [SET CALCPARALLEL](#)
- [the section called “CALCTASKDIMS”](#)
- [the section called “CALCPARALLEL”](#)
- [SET Commands](#)

SET CCTRACKCALC

Specifies whether Essbase checks the flags set by the [“CCTRACK” on page 408](#) setting to determine if the currency data has already been converted.

By default CCTRACK is turned on. Essbase tracks which currency partitions have been converted and which have not. The tracking is done at the currency partition level: a database with two partitions would have two flags that could be either "converted" or "unconverted." Essbase does not store a flag for member combinations within a partition.

When you load or clear data in a currency partition, Essbase does not reset the CCTRACK flag to "unconverted". You can use the SET CCTRACKCALC OFF command to force the conversion of the reloaded data, ignoring the CCTRACK flag.

Syntax

```
SET CCTRACKCALC ON | OFF;
```

Parameter Description

ON	Uses the flags set by the CCTRACK setting to determine whether the data needs to be converted. The default value is ON.
OFF	Always converts the data, regardless of whether CCTRACK has flagged the data as already-converted. Note that during the conversion CCTRACK is still active and tracks the exchange rates used during the conversion.

Notes

The SET CCTRACKCALC command is valid only when CCTRACK is set to TRUE (the default).

Example

```
SET CCTRACKCALC OFF;  
FIX (Actual)  
CCONV "XchR" ->US$;  
ENDFIX  
CALC ALL;
```

Fixes on the the Actual currency partition and forces the conversion of the Actual data regardless of whether Essbase has flagged the data as already being converted, converting the data using the XchR, US\$ rate from the currency database. Recalculates the database.

See Also

- [“CCTRACK” on page 408](#)
- [CLEARCCTRACK](#)
- [CCONV](#)
- [SET UPTOLOCAL](#)

SET CLEARUPDATESTATUS

Specifies when Essbase marks data blocks as clean. This clean status is used during Intelligent Calculation.

Syntax

```
SET CLEARUPDATESTATUS AFTER | ONLY | OFF;
```

Parameter Description

AFTER	Essbase marks calculated data blocks as clean, even if you are calculating a subset of your database.
ONLY	Essbase marks the specified data blocks as clean but does not actually calculate the data blocks. This does the same as AFTER, but disables calculation.
OFF	Essbase does not mark the calculated data blocks as clean. Data blocks are not marked as clean, even on a default calculation (<code>CALC ALL;</code>) of your database. The existing clean or dirty status of the calculated data blocks remains unchanged.

Notes

SET CLEARUPDATESTATUS specifies when Essbase marks data blocks as clean.

The data blocks in your database have a calculation status of either clean or dirty. When Essbase does a full calculation of your database, it marks the calculated data blocks as clean. When a data block is clean, Essbase will not recalculate the data block on subsequent calculations, provided that Intelligent Calculation is turned on.

To ensure the accuracy of your calculation results, consider carefully the effect of the SET CLEARUPDATESTATUS AFTER command on your calculation. .

If you do *not* use SET CLEARUPDATESTATUS, Essbase does *not* mark calculated data blocks as clean when you calculate a subset of your database. Essbase marks data blocks as clean only on a full calculation (`CALC ALL;`) or when Essbase calculates all members in a single calculation pass through your database.

If you calculate a subset of your database, you may want to use the SET CLEARUPDATESTATUS AFTER command to ensure that the calculated blocks are marked as clean. However, consider carefully the effect of this command on your calculation to ensure that your calculation results are correct.

Warnings

When you use the SET CLEARUPDATESTATUS command to mark calculated data blocks as clean, consider carefully the following questions:

Which data blocks are calculated?

Only *calculated* data blocks will be marked as clean.

Are concurrent calculations going to affect the same data blocks?

Do not use the SET CLEARUPDATESTATUS AFTER command with concurrent calculations unless you are certain that the different calculations will not need to calculate the same data block or blocks. If concurrent calculations attempt to calculate the same data blocks, with Intelligent Calculation turned on, Essbase may not recalculate the data blocks, because they are already marked as clean.

Are the same data blocks to be recalculated on a second calculation pass through the database?

If you calculate data blocks on a first calculation pass through your database, Essbase marks them as clean. If you then attempt to calculate the same data blocks on a subsequent pass with Intelligent Calculation turned on, Essbase does not recalculate the data blocks, because they are already marked as clean.

Example

The following examples are based on the Sample Basic database. They assume that Intelligent Calculation is turned on (the default). For information on turning Intelligent Calculation on and off, see the [SET UPDATECALC](#) command.

Example 1

```
SET CLEARUPDATESTATUS AFTER;
FIX ("New York")
CALC DIM(Product);
ENDFIX
```

New York is a member on the sparse Market dimension. Essbase searches for dirty parent data blocks for New York (for example "New York"->Colas in which Colas is a parent member). It calculates these dirty blocks based on the Product dimension and marks them as clean. Essbase does not mark the child, Input blocks as clean, because they are not calculated.

Example 2

```
SET CLEARUPDATESTATUS ONLY;
CALC ALL;
```

Essbase searches for all the dirty blocks in the database and marks them as clean. It does *not* calculate the blocks, even though a `CALC ALL;` command is used.

Example 3

```
SET CLEARUPDATESTATUS ONLY;
FIX ("New York")
CALC DIM(Product);
ENDFIX
```

New York is a member on the sparse Market dimension. Essbase searches for dirty parent data blocks for New York (for example "New York"->Colas in which Colas is a parent member). It marks them as clean. It does *not* calculate the data blocks. It does not mark the child blocks as clean because they are not calculated. For example, if

```
"New York"->100-10
```

is dirty, it remains dirty.

Example 4

```
SET CLEARUPDATESTATUS OFF;
CALC ALL;
CALC TWOPASS;
```

Essbase calculates all the dirty data blocks in the database. The calculated data blocks remain dirty; Essbase does *not* mark them as clean. Essbase then calculates those members tagged as

Two-Pass on the dimension tagged as Accounts. Again, it does not mark the calculated data blocks as clean.

See Also

- [SET UPDATECALC](#)
- [“UPDATECALC” on page 513](#)
- [SET Commands](#)

SET COPYMISSINGBLOCK

Sets whether the DATACOPY calculation command creates #MISSING blocks during the copy of data from a dense dimension.

This setting does not apply to aggregate storage databases.

SET COPYMISSINGBLOCK allows DATACOPY to avoid creating #MISSING blocks during the copy of data from a dense dimension.

Using DATACOPY on a dense dimension can create blocks populated with #MISSING. This is done deliberately in some instances, because most batch calculations operate only on existing data blocks. Therefore, DATACOPY can be used to ensure that all necessary data blocks are created prior to batch calculation.

But if the creation of #MISSING blocks is not required, you may want to avoid the increase in database size, and the possibly slower performance that results when, for example, a default calculation visits every #MISSING block.

Syntax

```
SET COPYMISSINGBLOCK ON | OFF
```

Parameter Description

ON	This is the default value. Allows missing blocks to be created during a data copy.
OFF	Suppresses the creation of missing blocks during a data copy.

Notes

- Existing #MISSING blocks are not removed.
- A message is added to the Essbase Server log to indicate the number of data blocks being copied from the source data blocks. The number of #MISSING blocks skipped, if any, is also reported in the log.

Example

```
SET COPYMISSINGBLOCK OFF;
```

The following log message indicates that SET COPYMISSINGBLOCK is OFF:

```
[Fri May 31 10:35:03 2002]Local/Test6/Test6/essexer/Info(1012574)  
Datacopy command copied [1] source data blocks to [0] target data blocks
```


See Also

- [DATACOPY](#)

SET CREATENONMISSINGBLK

Controls whether potential blocks are created in memory for calculation purposes, and whether #MISSING blocks are stored. It affects the results of calculations on sparse and dense dimensions.

By default, Essbase applies dense-member formulas only to existing data blocks. SET CREATENONMISSINGBLK ON enables Essbase to create potential blocks in memory where the dense-member formulas are performed. Of these potential blocks, Essbase writes to the database only blocks that contain values; blocks resulting in only #MISSING are not written to the database.

The creation of #MISSING blocks resulting from sparse-member formulas is governed by the Create Block on Equations setting. (See [SET CREATEBLOCKONEQ](#).) The SET CREATENONMISSINGBLK ON command ensures that only non-empty blocks are created, regardless of the Create Block on Equations setting.

In order to create new blocks, setting SET CREATENONMISSINGBLK to ON requires Essbase to anticipate the blocks that will be created. Working with potential blocks can affect calculation performance. Consider the following situations carefully:

- When SET CREATENONMISSINGBLK is ON, all sparse-member formulas are executed in top-down mode. Dense member formulas are flagged for top-down calculation when they contain the following:
 - Sparse members
 - Constants (for example, Sales = 100,000)
 - The @VAR function
 - The @XREF function
- If Essbase encounters the [@CALCMODE\(BOTTOMUP\)](#) in a member formula, it ignores the @CALCMODE command. A message about the member is written in the application log saying that the command is being ignored.
- If a batch calculation contains top-down formulas and SET CREATENONMISSINGBLK is ON, Intelligent Calculation is turned off. Within the scope of the calculation script, all blocks are calculated, regardless if they are marked clean or dirty.
- To reduce the number of blocks to be calculated, use this command within FIX/ENDFIX regions. As a warning, when the potential number of blocks exceeds 20 million, Essbase writes an entry to the application log showing the number of blocks to be calculated and recommending using FIX/ENDFIX.
- You can use multiple SET CREATENONMISSINGBLK commands in a calc script, each affecting calculations that follow. However, consider that each time SET

CREATENONMISSINGBLK is encountered within a set of FIX and ENDFIX statements, the calculator cycles through the database, potentially affecting calculation performance.

Syntax

```
SET CREATENONMISSINGBLK ON|OFF;
```

Parameter Description

- | | |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ON | Calculations are performed on potential blocks as well as existing blocks. If the result of the calculation is not #MISSING, the block is stored. The Create Blocks on Equations setting is ignored. |
| OFF | Calculations are performed only on existing blocks. This is the default setting. |

Notes

- SET CREATENONMISSINGBLK affects only creation of new blocks. If existing blocks become #MISSING after formula execution, they are not deleted.
- The value set by SET CREATENONMISSINGBLK stays in effect until the next SET CREATENONMISSINGBLK is processed, or the calculation script terminates.
- When the calculation script includes both SET CREATENONMISSINGBLK ON and SET MSG DETAIL, any non-stored #MISSING block is indicated in the application log.
- If SET MSG is set to SUMMARY, when SET CREATENONMISSINGBLK is set to ON, Essbase writes an entry to the application log stating that Create Non #MISSING Blocks is enabled.
- If SET MSG is set to SUMMARY, and SET CREATENONMISSINGBLK is set to ON, at the end of the calculation, Essbase writes an entry to the application log showing the total number of #MISSING blocks that were not created.

Example

The following example is based on a variation of Sample Basic. Assume that the Scenario dimension, of which Actual is a member, is sparse. "Jan Rolling YTD Est" is a member of the dense time dimension, Year.

```
FIX (Budget)
  SET MSG DETAIL;
  SET CREATENONMISSINGBLK ON;
  "Jan Rolling YTD Est" = (Jan->Actual+Feb+Mar+Apr+May+Jun+Jul+Aug+Sep+Oct+Nov+Dec);
ENDFIX
```

See Also

- [SET CREATEBLOCKONEQ](#)

SET CREATEBLOCKONEQ

Controls, within a calculation script, whether new blocks are created when a calculation formula assigns anything other than a constant to a member of a sparse dimension. SET CREATEBLOCKONEQ overrides the Create Block on Equation setting for the database.

Syntax

```
SET CREATEBLOCKONEQ ON|OFF;
```

Parameter Description

ON	When a formula assigns a non-constant value to a sparse dimension member for which no block exists, Essbase creates a block.
OFF	When a formula assigns a non-constant value to a sparse dimension member for which no block exists, Essbase does not create a block.

Notes

If calculations result in a value for a sparse dimension member for which no block exists, Essbase creates a block. Sometimes, new blocks are not desired; for example, when they contain no other values. In large databases, creation and processing of unneeded blocks can increase processing time and storage requirements.

The Create Blocks on Equation setting is designed for situations when blocks would be created as a result of assigning something other than a constant to a member of a sparse dimension. For example, when Create Blocks on Equation is ON and West is assigned a value where it did not have a value before, new blocks are created. When this setting is OFF, blocks are not created.

Create Blocks on Equation setting is a database property. Its initial value is OFF; no blocks are created when something other than a constant is assigned to a sparse dimension member. Use Administration Services or MaxL to turn the setting ON at the database-level. For more information about enabling Create Blocks on Equation, see the MaxL documentation in the *Oracle Essbase Technical Reference* or the *Oracle Essbase Administration Services Online Help*.

For more specific control, you can use the SET CREATEBLOCKONEQ calculation command within a calculation script to control creation of blocks at the time the command is encountered in the script. Use of SET CREATEBLOCKONEQ has the following characteristics:

- When Essbase encounters SET CREATEBLOCKONEQ within a calculation script, the database-level setting is ignored.
- You can use multiple SET CREATEBLOCKONEQ commands in the calculation script to define the Create Blocks on Equation setting value for the calculations following each command.
- The value set by the SET CREATEBLOCKONEQ command stays in effect until the next SET CREATEBLOCKONEQ command is processed or the calculation script is finished.
- The Create Blocks on Equation setting is overridden by SET CREATENONMISSINGBLK ON (see [SET CREATENONMISSINGBLK](#)).
- The SET CREATEBLOCKONEQ command does not change the database-level Create Blocks on Equation property.
- If no SET CREATEBLOCKONEQ command is encountered, Essbase uses the database-level setting to determine whether to create blocks.

When the Create Blocks on Equation setting is ON, Essbase uses the top-down calculation method to calculate each sparse member.

The Create Blocks on Equation setting is not consulted when Essbase assigns constants to members of sparse dimensions. The following table shows examples of sparse member calculations where constants or non-constants are assigned to them.

Assigned Value	Sparse Member Formula Example	New Block Created?
Constant	West = 350	Yes
Non-constant	West = California + 120	Yes, if the Create Blocks on Equation setting is ON. Otherwise, no.
Non-constant	West = California * 1.05	Yes, if the Create Blocks on Equation setting is ON. Otherwise, no.

For a tip on controlling creation of blocks when you work with non-constants and sparse dimensions, in the *Oracle Essbase Database Administrator's Guide* check for information about improving performance for non-constants assigned to members in sparse dimensions.

Example

The following example is based on Sample.Basic. West and East are members of the sparse Markets dimension.

```
FIX (Colas);
SET CREATEBLOCKONEQ OFF
West = California + 120;
SET CREATEBLOCKONEQ ON
East = "New York" + 100;
ENDFIX
```

Because of the preceding SET CREATEBLOCKONEQ OFF command, Essbase does not create blocks for new values of West. Because the setting has been reversed to ON in the next command, Essbase creates blocks for new values of East.

See Also

- [SET CREATENONMISSINGBLK](#)

SET DATAEXPORTOPTIONS

Specifies options for data export operations.

Syntax

```
SET DATAEXPORTOPTIONS
{
  DataExportLevel ALL | LEVEL0 | INPUT;
  DataExportDynamicCalc ON | OFF;
  DataExportNonExistingBlocks ON | OFF;
  DataExportDecimal n;
  DataExportPrecision n;
  DataExportColFormat ON | OFF;
  DataExportColHeader dimensionName;
  DataExportDimHeader ON | OFF;
  DataExportRelationalFile ON | OFF;
  DataExportOverwriteFile ON | OFF;
```

```
DataExportDryRun ON | OFF;  
};
```

Notes

Each SET DATAEXPORTOPTIONS command specifies a set of option values that are in place until the next SET DATAEXPORTOPTIONS command is encountered. At that time, option values are reset to default and newly specified option values are set.

The option list must start with a left brace ({) and end with a right brace followed by a semicolon (} ;). Each option ends with a semicolon (;). The options can be listed in any order. When an option is not specified, the default value is assumed.

The options are described here in three categories:

- [Content Options](#)
- [Output Format Options](#)
- [Processing Options](#)

Content Options

```
DataExportLevel ALL | LEVEL0 | INPUT
```

- **All**—(Default) All data, including consolidation and calculation results.
- **Level 0**—Data from level 0 data blocks only (blocks containing only level 0 sparse member combinations).
- **Input**—Input blocks only (blocks containing data from a previous data load or spreadsheet Lock & Send). This option excludes dynamically calculated data. See also the [DataExportDynamicCalc](#) option.

Description

Specifies the amount of data to export.

```
DataExportDynamicCalc ON | OFF
```

- **ON**—(Default) Dynamically calculated values are included in the export.
- **OFF**—No dynamically calculated values are included in the report.

Description

Specifies whether a text data export excludes dynamically calculated data.

Notes:

- Text data exports only. If DataExportDynamicCalc ON is encountered with a binary export (DATAEXPORT BINFILE ...) it is ignored. No dynamically calculated data is exported.
- The DataExportDynamicCalc option does not apply to attribute values.
- If [DataExportLevel](#) INPUT is also specified and the FIX statement range includes sparse Dynamic Calc members, the FIX statement is ignored.

```
DataExportNonExistingBlocks ON | OFF
```

- **ON**—Data from all possible data blocks, including all combinations in sparse dimensions, are exported.
- **OFF**—(Default) Only data from existing data blocks is exported.

Description

Specifies whether to export data from all possible data blocks. For large outlines with a large number of members in sparse dimensions, the number of potential data blocks can be very high. Exporting Dynamic Calc members from all possible blocks can significantly impact performance.

DataExportPrecision *n*

n(Optional, default 16)—A value that specifies the number of positions in exported numeric data. If *n* < 0, 16-position precision is used.

Description

Specifies that the DATAEXPORT calculation command will output numeric data with emphasis on precision (accuracy). Depending on the size of a data value and number of decimal positions, some numeric fields may be written in exponential format; for example, 678123e+008. You may consider using DataExportPrecision for export files intended as backup or when data ranges from very large to very small values. The output files typically are smaller and data values more accurate. For output data to be read by people or some external programs, you may consider specifying the [DataExportDecimal](#) option instead.

Notes:

- By default, Essbase supports 16 positions for numeric data, including decimal positions.
- The [DataExportDecimal](#) option has precedence over the DataExportPrecision option.

Example

```
SET DATAEXPORTOPTIONS
{
  DataExportPrecision 6;
  DataExportLevel "ALL";
  DataExportColHeader "Measures";
  DataExportDynamicCalc ON;
};
DATAEXPORT "File" ", " "output1.out";
```

Initial Data Load Values

```
"Sales" "COGS" "Margin" "Marketing" "Payroll" "Misc" "Total Expenses" "Profit" "Opening
Inventory" "Additions" "Ending Inventory" "Margin %" "Profit %"
"100-10" "New York"
"Jan" "Actual" 678123456.0 271123456.0 407123456.0 941234567890123456.0 51123456.0 0
145123456.0 262123456.0 2101123456.0 644123456.0 2067123456.0 60123456.029 38123456.6430
"Feb" "Actual" 645123 258123 3871234 9012345 5112345 112345678 14212345 24512345
2067123456 61912345 20411234 601234 37123456.98
"Mar" "Actual" 675 270 405 94 51 1 146 259 2041 742 2108 60 38.37037037037037
"Qtr1" "Actual" 1998 799 1199 278 153 2 433 766 2101 2005 2108 60.01001001001001 38.
33833833833834
```

Exported Data Format

```
"Sales", "COGS", "Margin", "Marketing", "Payroll", "Misc", "Total Expenses", "Profit", "Opening
Inventory", "Additions", "Ending Inventory", "Margin %", "Profit %", "Profit per
Ounce", "100-10", "New York"
"Jan", "Actual", 6.78123e+008, 2.71123e+008, 4.07e+008, 9.41235e+017, 5.11235e+007, 0, 9.41235e
+017, -9.41235e+017, 2.10112e+009, 6.44123e+008, 2.06712e+009, 60.0186, -1.388e+011, -7.84362e
+016
"Feb", "Actual", 645123, 258123, 387000, 9.01235e+006, 5.11235e+006, 1.12346e+008, 1.2647e
+008, -1.26083e+008, 2.06712e+009, 6.19123e+007, 2.04112e+007, 59.9886, -19544.1, -1.05069e+007
"Mar", "Actual", 675, 270, 405, 94, 51, 1, 146, 259, 2041, 742, 2108, 60, 38.3704, 21.5833
```

DataExportDecimal *n*

Where *n* is a value between 0 and 16.

If no value is provided, the number of decimal positions of the data to be exported is used, up to 16 positions, or a value determined by the [DataExportPrecision](#) option if that is specified.

Description

Specifies that the DATAEXPORT calculation command will output numeric data with emphasis on legibility; output data is in straight text format. Regardless of the number of decimal positions in the data, the specified number is output. It is possible the data can lose accuracy, particularly if the data ranges from very large values to very small values, above and below the decimal point.

Notes:

- By default, Essbase supports 16 positions for numeric data, including decimal positions.
- If both the DataExportDecimal option and the [DataExportPrecision](#) option are specified, the DataExportPrecision option is ignored.

Example

```
SET DATAEXPORTOPTIONS
  {DataExportDecimal 4;
  DataExportLevel "ALL";
  DataExportColHeader "Measures";
  DataExportDynamicCalc ON;
  };
DATAEXPORT "File" ", " "output1.out";
```

Initial Data Load Values

```
"Sales" "COGS" "Margin" "Marketing" "Payroll" "Misc" "Total Expenses" "Profit" "Opening
Inventory" "Additions" "Ending Inventory" "Margin %" "Profit %"
"100-10" "New York"
"Jan" "Actual" 678123456.0 271123456.0 407123456.0 941234567890123456.0 51123456.0 0
145123456.0 262123456.0 2101123456.0 644123456.0 2067123456.0 60123456.029 38123456.6430
"Feb" "Actual" 645123 258123 3871234 9012345 5112345 112345678 14212345 24512345
2067123456 61912345 20411234 601234 37123456.98
"Mar" "Actual" 675 270 405 94 51 1 146 259 2041 742 2108 60 38.37037037037037
"Qtr1" "Actual" 1998 799 1199 278 153 2 433 766 2101 2005 2108 60.01001001001001 38.
33833833833834
```

Exported Data Format

```
"Sales", "COGS", "Margin", "Marketing", "Payroll", "Misc", "Total Expenses", "Profit", "Opening
Inventory", "Additions", "Ending Inventory", "Margin %", "Profit %", "Profit per Ounce"
"100-10", "New York"
"Jan", "Actual", 678123456.0000, 271123456.0000, 407000000.0000, 941234567890123520.0000,
51123456.0000, 0.0000, 941234567941246980.0000, -941234567534246910.0000, 2101123456.0000,
644123456.0000, 2067123456.0000, 60.0186, -138799883591.4395, -78436213961187248.0000
"Feb", "Actual", 645123.0000, 258123.0000, 387000.0000, 9012345.0000, 5112345.0000, 112345678.
0000, 126470368.0000, -126083368.0000, 2067123456.0000, 61912345.0000, 20411234.0000, 59.
9886, -19544.0820, -10506947.3333
"Mar", "Actual", 675.0000, 270.0000, 405.0000, 94.0000, 51.0000, 1.0000, 146.0000, 259.0000, 2041.
0000, 742.0000, 2108.0000, 60.0000, 38.3704, 21.5833
```

Output Format Options

DataExportColFormat ON | OFF

- ON—The data is output in columnar format.
- OFF—Default. The data is output in non-columnar format.

Description

Specifies if data is output in columnar format. Columnar format displays a member name from every dimension; names can be repeated from row to row, enabling use by applications other than Essbase tools. In non-columnar format, sparse members identifying a data block are included only once for the block. Non-columnar export files are smaller, enabling faster loading to an Essbase database.

Notes

- Do not use the DataExportColFormat option in combination with the [DataExportRelationalFile](#) option, which already assumes columnar format for files destined as input files to relational databases.

Example

```
SET DATAEXPORTOPTIONS
{
  DATAEXPORTCOLFORMAT ON;
};
FIX("100-10", Sales, COGS, Jan, Feb, Mar, Actual, Budget)
  DATAEXPORT "File" ", " "d:\temp\test2.txt" ;
ENDFIX;
```

DataExportColHeader *dimensionName*

Description

Specifies the name of the dense dimension that is the column header (the focus) around which other data is referenced in the export file. Use the DataExportColHeader option only when you export data to a text file. For example, if from Sample Basic the Year dimension is specified, the output data starts with data associated with the first member of the Year dimension: Year. After all data for Year is output, it continues with the second member: Qtr1, and so on.

Notes

- MaxL, ESSCMD, and Essbase exports do not provide a similar capability. With these methods, Essbase determines the focal point of the output data.

Exporting through Report Writer enables you to specify the header in the report script.

Example

```
SET DATAEXPORTOPTIONS {DATAEXPORTCOLHEADER Scenario;};
```

Specifies Scenario as the page header in the export file. The Scenario dimension contains three members: Scenario, Actual, and Budget. All Scenario data is shown first, followed by all Actual data, then all Budget data.

DataExportDimHeader ON | OFF

- ON—The header record is included.
- OFF—Default. The header record is not included.

Description

Use the DataExportDimHeader option to insert the optional header record at the beginning of the export data file. The header record contains all dimension names in the order as they are used in the file. Specifying this command always writes the data in "column format".

Example

```
SET DATAEXPORTOPTIONS
{
  DATAEXPORTLEVEL "ALL";
  DATAEXPORTDIMHEADER ON;
};
FIX("100-10", "New York", "Actual")
  DATAEXPORT "File" ",," "E:\temp\2222.txt" ;
ENDFIX;
```

Specifying the DataExportDimHeader ON option while exporting Sample Basic writes the data in column format, with common members repeated in each row. The data begins with a dimension header, as shown in the first two rows of the example file below:

```
"Product", "Market", "Year", "Scenario", "Measures"
"Sales", "COGS", "Marketing", "Payroll", "Misc", "Opening Inventory", "Additions", "Ending
Inventory"
"100-10", "New York", "Jan", "Actual", 678,271,94,51,0,2101,644,2067
"100-10", "New York", "Feb", "Actual", 645,258,90,51,1,2067,619,2041
"100-10", "New York", "Mar", "Actual", 675,270,94,51,1,2041,742,2108
"100-10", "New York", "Apr", "Actual", 712,284,99,53,0,2108,854,2250
"100-10", "New York", "May", "Actual", 756,302,105,53,1,2250,982,2476
"100-10", "New York", "Jun", "Actual", 890,356,124,53,0,2476,1068,2654
"100-10", "New York", "Jul", "Actual", 912,364,127,51,0,2654,875,2617
"100-10", "New York", "Aug", "Actual", 910,364,127,51,0,2617,873,2580
"100-10", "New York", "Sep", "Actual", 790,316,110,51,1,2580,758,2548
"100-10", "New York", "Oct", "Actual", 650,260,91,51,1,2548,682,2580
"100-10", "New York", "Nov", "Actual", 623,249,87,51,0,2580,685,2642
"100-10", "New York", "Dec", "Actual", 699,279,97,51,1,2642,671,2614
```

DataExportRelationalFile ON | OFF

- ON—The output text export file is formatted for import to a relational database.
 - Data is in column format; sparse member names are repeated. (The [DataExportColFormat](#) option is ignored.)
 - The first record in the export file is data; no column heading or dimension header is included, even if specified. (The [DataExportColHeader](#) and [DataExportDimHeader](#) options are ignored.)
 - Missing and invalid data is skipped, resulting in consecutive delimiters (commas) in the output. The optional "missing_char" parameter for [DATAEXPORT](#) is ignored
- OFF—Default. The data is not explicitly formatted for use as input to a relational database.

Description

Using the [DataExportRelationalFile](#) option with [DATAEXPORT](#) enables you to format the text export file to be used directly as an input file for a relational database.

Example

```
SET DATAEXPORTOPTIONS {
  DataExportLevel "ALL";
  DataExportRelationalFile ON;
};

FIX (Jan)
  DATAEXPORT "File" ", " c:\monthly\jan.txt
ENDFIX;
```

Processing Options

[DataExportOverwriteFile](#) ON | OFF

- ON—The existing file with the same name and location is replaced.
- OFF—Default. If a file with the same name and location already exists, no file is output.

Description

Manages whether an existing file with the same name and location is replaced.

[DataExportDryRun](#) ON | OFF

- ON—[DATAEXPORT](#) and associated commands are run, without exporting data.
- OFF—Default. Data is exported

Description

Enables running the calculation script data export commands to see information about the coded export, without exporting the data. When the [DataExportDryRun](#) option value is ON, the following information is written to the output file specified in the [DATAEXPORT](#) command:

- Summary of data export settings
- Info, Warning, and Error messages
- Exact number of blocks to be exported

- Estimated time, excluding I/O time.

Notes

- The DataExportDryRun option does not work with exports to relational databases.
- If you modify the script for reuse for the actual export, besides removing the DataExportDryRun option from the script you may want to change the name of the export file.

Example

```
SET DATAEXPORTOPTIONS
{
DataExportLevel "ALL";
DataExportColHeader "Measures";
DataExportColFormat ON;
DataExportDimHeader ON;
DataExportDynamicCalc OFF;
DataExportDecimal 0;
DataExportDryRun ON;
DataExportOverwriteFile ON;
};

FIX("Qtr1")
DATAEXPORT "File" ", " "E:\temp\log.txt" ;
ENDFIX;
```

Creates the file "E:\temp\log.txt" containing the following information:

```
<EXPORT_OPTIONS>
  <DELIMITER>
  '
  </DELIMITER>
  <MISSING_VALUE>
  #Mi
  </MISSING_VALUE>
  <EXPORT_LEVEL>
  ALL
  </EXPORT_LEVEL>
  <DYNAMIC_CALC_EXPORT>
  OFF
  </DYNAMIC_CALC_EXPORT>
  <COLUMN_HEADER>
  Measures
  </COLUMN_HEADER>
  <COLUMN_FORMAT>
  ON
  </COLUMN_FORMAT>
  <DIMENSION_HEADER_WRITE>
  ON
  </DIMENSION_HEADER_WRITE>
  <FILE_OVERWRITE>
  ON
  </FILE_OVERWRITE>
  <DECIMAL_POINT>
  ON
```

```

</DECIMAL_POINT>
<PRECISION_POINT>
16
</PRECISION_POINT>
<RELATIONAL_EXPORT>
OFF
</RELATIONAL_EXPORT>
</EXPORT_OPTIONS>
<MESSAGE>
  <INFO>
    DataExport Warning: FIX statement contains Dynamic Calc member [Qtr1]. No
Dynamic Calc members are exported with the DataExportDynamicCalc option set to OFF.
  </INFO>
  <INFO>
    Data Export Completed. Total blocks: [332]. Elapsed time: [3.846] secs.
  </INFO>
</MESSAGE>

```

See Also

- [DATAEXPORT](#)
- [FIX...ENDFIX](#)
- [SET Commands](#)

SET DATAIMPORTIGNORETIMESTAMP

Specifies whether to ignore the outline timestamp captured at the time the data was exported.

Syntax

```
SET DATAIMPORTIGNORETIMESTAMP ON|OFF;
```

Parameter Description

ON	Ignore the outline timestamp.
OFF	Default. Check the outline timestamp.

Notes

The [DATAEXPORT](#) "Binfile" command captures the outline timestamp when it creates a binary export file. By default, when the file is imported, Essbase checks the import file timestamp against the existing outline timestamp to ensure the correct import file is read. You can use SET DATAIMPORTIGNORETIMESTAMP to bypass checking the timestamp.

Caution! Bypassing the check enables potentially importing the wrong file.

Example

```
SET DATAIMPORTIGNORETIMESTAMP ON;
DATAIMPORTBIN e:january\basic.bin
```

Specifies to ignore comparing the outline timestamp with the timestamp on the import tile, and to import the binary export file to the database on which the calculation script is running.

See Also

- [DATAEXPORT](#)
- [DATAIMPORTBIN](#)
- [SET Commands](#)

SET EMPTYMEMBERSETS

EMPTYMEMBERSETS stops the calculation within a [FIX...ENDFIX](#) command if the FIX evaluates to an empty member set.

Syntax

```
SET EMPTYMEMBERSETS ON|OFF
```

Parameter Description

ON	Calculation within FIX command stops if FIX evaluates to an empty member set.
OFF	Entire database is calculated, even if FIX evaluates to an empty member set.

Notes

If EMPTYMEMBERSETS is ON, and a FIX command evaluates to a empty member set, the calculation within the FIX command stops and the following information message is displayed: "FIX statement evaluates to an empty set. Please refer to SET EMPTYMEMBERSETS command." The calculation resumes after the FIX command. If a calculation script contains nested FIX commands, the nested FIX commands are not evaluated.

Example

The following calculation script does not calculate Calc Dim(Year) within the FIX command. 100-10 has no children and therefore the FIX statement evaluates to an empty member set.

```
SET EMPTYMEMBERSETS ON;
...
FIX(@CHILDREN("100-10"))
  Calc Dim(Year);
ENDFIX
...
```

The following calculation script has nested FIX commands. Calc Dim(Product) is not calculated because FIX(@CHILDREN("100-10")) evaluates to empty member set. Calc Dim(Year) is not calculated even though the nested FIX("New York") does not evaluate to an empty member set.

```
SET EMPTYMEMBERSETS ON;
...
FIX(@CHILDREN("100-10"))
  FIX("New York")
    Calc Dim(Year);
  ENDFIX
Calc Dim (Product);
```

ENDFIX

...

SET FRMLBOTTOMUP

Optimizes the calculation of complex formulas on sparse dimensions in large database outlines. This command tells Essbase to perform a bottom-up calculation on formulas that would otherwise require a top-down calculation.

You might want to turn on this setting when using the `CALC ALL` and `CALC DIM` commands to calculate the database.

Syntax

```
SET FRMLBOTTOMUP ON|OFF;
```

Parameter Description

ON	Turns on the bottom-up sparse formula calculation method.
OFF	Turns off the bottom-up sparse formula calculation method. The default setting is OFF. You can change this setting by using <code>CALCOPTFRMLBOTTOMUP TRUE</code> in the <code>essbase.cfg</code> file.

Notes

- For information on complex formulas and top-down calculations, see the *Oracle Essbase Database Administrator's Guide*.
- Forcing a bottom-up calculation on a formula may produce results that are inconsistent with a top-down calculation if:
 - The formula contains complex functions (for example, range functions)
 - The formula's dependencies are not straightforward
- Before using the `SET FRMLBOTTOMUP` command in a production environment, be sure to check the validity of calculation results produced when the command is enabled (set to ON).

Example

```
SET FRMLBOTTOMUP ON;
```

See Also

- [the section called “CALCOPTFRMLBOTTOMUP”](#)
- [SET Commands](#)

SET FRMLRTDYNAMIC

Enables you to turn off calculation of all dense Dynamic Calc members during batch calculation if runtime dependent functions are included in formulas on stored members. (The preprocessing phase of a calculation script cannot determine if an outline contains dense Dynamic Calc members.)

This command improves batch calculation performance by removing the overhead of calculating all Dynamic Calc members.

The SET FRMLRTDYNAMIC command can be applied to an entire calculation script segment, as shown in the example below.

Syntax

```
SET FRMLRTDYNAMIC ON | OFF;
```

Parameter Description

ON Calculation of Dynamic Calc members is performed. The default value is ON.

OFF Calculation of Dynamic Calc members is not performed.

Notes

- Runtime-dependent functions include:
 - @ANCEST
 - @SANCEST
 - @PARENT
 - @SPARENT
 - @CURRMBR
- If a stored member formula includes a runtime-dependent function on a Dynamic Calc member, it may get #MISSING as the result instead of the expected value after executing the formula on the Dynamic Calc member.

Example

The following example turns off all dense Dynamic Calc members:

```
SET FRMLRTDYNAMIC OFF;  
FIX(@LEVMBRS(Product, 0))  
"Avg Sales" = @AVGRANGE(SKIPNONE, Sales, @CHIDREN(@CURRMBR(Product)));  
ENDFIX  
CALC ALL;
```

SET LOCKBLOCK

Specifies the maximum number of blocks that Essbase can get addressability to concurrently when calculating a sparse member formula.

You can choose one of three levels. The number of blocks that Essbase can get addressability to at each level is defined using the CALCLOCKBLOCK setting in the `essbase.cfg` file.

Syntax

```
SET LOCKBLOCK HIGH | DEFAULT | LOW;
```

Parameter	Description
HIGH, DEFAULT, and LOW	Levels defining the number of blocks that Essbase can get addressability to concurrently.

Notes

When a block is calculated, Essbase locks (gets addressability to) the block along with the blocks containing its children. Essbase calculates the block and then releases it along with the blocks containing its children.

By default, Essbase allows up to 100 blocks to be locked (addressable) concurrently when calculating a block. This is sufficient for most database calculations.

However, you may want to set a number higher than 100 if you are consolidating very large numbers of children in a formula calculation. This setting ensures that Essbase can get addressability to all the required blocks when calculating a data block and that performance will not be impaired.

For more information on data blocks, see the *Oracle Essbase Database Administrator's Guide*.

Example

If the `essbase.cfg` file contains the following settings:

```
CALCLOCKBLOCKHIGH          500
CALCLOCKBLOCKDEFAULT      200
CALCLOCKBLOCKLOW          50
```

then:

```
SET LOCKBLOCK HIGH;
```

means that Essbase can get addressability to up to 500 data blocks when calculating one block.

```
SET LOCKBLOCK DEFAULT;
```

means that Essbase can get addressability to up to 200 data blocks when calculating one block.

```
SET LOCKBLOCK LOW;
```

means that Essbase can get addressability to up to 50 data blocks when calculating one block.

See Also

- [the section called “CALCLOCKBLOCK”](#)
- [SET Commands](#)

SET MSG

Sets the level of messaging you want returned about calculations, and enables simulated calculations.

The SET MSG command applies only to the calculation script in which it is used.

Syntax

```
SET MSG SUMMARY | DETAIL | ERROR | INFO | NONE | ONLY;
```


Parameter	Description
-----------	-------------

SUMMARY	Displays calculation settings and provides statistics on the number of: <ul style="list-style-type: none">● Data blocks created, read, and written● Data cells calculated
DETAIL	Provides the same information as SUMMARY. In addition, it displays a detailed information message every time Essbase calculates a data block.
ERROR	Displays only error messages.
INFO	Displays information and error messages.
NONE	Displays no messages during the life of the calculation script. However, because error messages may contain vital information, they are still displayed.
ONLY	Instructs Essbase to perform a simulated calculation only. You may disregard any error message during validation that indicates Essbase does not recognize a command.

Note: When you use this parameter, Essbase generates some empty upper-level blocks. Make sure to clear upper-level blocks (or non-input blocks if you load data into upper level blocks in your model) at the end of the simulation/command.

We recommend using SET MSG ONLY with the calculation script commands SET NOTICE HIGH and CALC ALL. For more information, see the *Oracle Essbase Database Administrator's Guide* sections on optimizing calculations.

SET MSG ONLY does not generate a completion notice.

Notes

SET MSG SUMMARY and SET MSG DETAIL tell you:

- The status of calculation settings (for example, whether completion notice messages are enabled)
- The total number of data blocks created
- The number of data blocks read and written on sparse calculations
- The number of data blocks read and written on dense calculations
- The number of data cells calculated on sparse calculations
- The number of data cells calculated on dense calculations

In addition, the SET MSG DETAIL command provides an information message every time Essbase calculates a data block. It is useful for testing your database's consolidation path. Because it causes a high processing overhead, it should be used during test calculations only.

SET MSG SUMMARY causes a processing overhead of approximately 1% to 5%, depending on the database size.

Example

```
SET MSG ERROR;
```

Displays only the error messages.

SET MSG SUMMARY;

Produces the following sample output:

```
[Tue Apr 4 05:11:16 1995] local/Sample/Basic/Qatest/Info(1012672)
Calculator Information Message:
```

```
Maximum Number of Lock Blocks: [100] Blocks
```

```
Completion Notice Messages: [Disabled]
```

```
Calculations On Updated Blocks Only: [Enabled]
```

```
Clear Update Status After Full Calculations: [Enabled]
```

```
Calculator Cache With Multiple Bitmaps For: [Market]
```

```
[Tue Apr 4 05:11:19 1995] local/Sample/Basic/Qatest/Info(1012672)
Calculator Information Message:
```

```
Total Block Created: [0.0000e+00] Blocks
```

```
Sparse Calculations: [4.3000e+01] Writes and [4.3000e+01] Reads
```

```
Dense Calculations: [4.3200e+02] Writes and [4.3200e+02] Reads
```

```
Sparse Calculations: [1.7200e+02] Cells
```

```
Dense Calculations: [4.3200e+02] Cells
```

SET MSG DETAIL;

Produces the following sample output:

```
[Thu Mar 30 16:27:26 1995] local/Sample/Basic/Qatest/Info(1012669)
Calculator Information Message:
```

```
Maximum Number of Lock Blocks: [100] Blocks
```

```
Completion Notice Messages: [Disabled]
```

```
Calculations On Updated Blocks Only: [Enabled]
```

```
Clear Update Status After Partial Calculations: [Disabled]
```

```
Calculator Cache With Multiple Bitmaps For: [Market]
```

```
[Thu Mar 30 16:27:26 1995] local/Sample/Basic/Qatest/Info(1012669)
Calculator Information Message: Executing Block - [100], [East]
```

```
[Thu Mar 30 16:27:26 1995] local/Sample/Basic/Qatest/Info(1012669)
```

```
Calculator Information Message: Executing Block - [Product], [East]
```

```
[Thu Mar 30 16:27:26 1995] local/Sample/Basic/Qatest/Info(1012669)
```

```

Calculator Information Message: Executing Block - [100], [Market]

[Thu Mar 30 16:27:26 1995] local/Sample/Basic/Qatest/Info(1012669)

Calculator Information Message: Executing Block - [Product], [Market]

[Thu Mar 30 16:27:26 1995] local/Sample/Basic/Qatest/Info(1012669)
Calculator Information Message:

Total Block Created: [0.0000e+00] Blocks

Sparse Calculations: [4.0000e+00] Writes and [2.2000e+01] Reads

Dense Calculations: [0.0000e+00] Writes and [0.0000e+00] Reads

Sparse Calculations: [3.8080e+03] Cells

Dense Calculations: [0.0000e+00] Cells

```

See Also

- [CLEARBLOCK](#)
- [SET NOTICE](#)
- [SET Commands](#)

SET NOTICE

Monitors the progress of your calculation by providing completion notices at intervals during the calculation. The number of notices depends on the level you specify.

Syntax

```
SET NOTICE HIGH | DEFAULT | LOW;
```

Parameter	Description
HIGH, DEFAULT, and LOW	Levels defining the frequency and number of completion notices. You can set the values of HIGH, DEFAULT, and LOW using the CALCNOTICE setting in the <code>essbase.cfg</code> file. If you do not set the value of DEFAULT in the <code>essbase.cfg</code> file, Essbase uses a default value of 10, which provides 10 completion messages at 10% intervals during the calculation.

Notes

- You can specify the number of notices for each level using the CALCNOTICE setting in the `essbase.cfg` file.
- The interval between notices is approximate. Essbase measures the interval by taking the number of data blocks already calculated as a percentage of the total number of possible data blocks in your database. For example, if there are 10,000 possible blocks and you specify 5 notices, Essbase notifies you when the calculation approximately reaches block 2000, 4000, 6000, 8,000 and 10,000. However, if only the blocks 1,000 - 4,000 exist, then Essbase displays only two notices.

- For partial calculations and calculations with multiple passes through your database, the interval between completion notices is very approximate.
- Completion notices do not significantly reduce the calculation performance, except when used with a very small database.

Example

If the `essbase.cfg` file contains the following settings:

```
CALCNOTICEHIGH 50
CALCNOTICEDEFAULT 20
CALCNOTICELOW 5
```

then:

```
SET NOTICE HIGH;
```

displays 50 completion notices at 2% intervals.

```
SET NOTICE DEFAULT;
```

displays 20 completion notices at 5% intervals.

```
SET NOTICE LOW;
```

displays 5 completion notices at 20% intervals.

```
SET NOTICE LOW;
```

might produce the following sample output:

```
[Thu Apr 6 10:09:19 1995] Local/Sample/Basic/Qatest/Info(1012669)
Calculating [ Measures(All members) Year(All members) Scenario(All members) Product(All
members) Market(All members) ]
```

```
[Thu Apr 6 10:09:19 1995] Local/Sample/Basic/Qatest/Info(1012672)
Calculator Information Message:
Maximum Number of Lock Blocks: [100] Blocks
Completion Notice For Every: [ 10.000%] Of Blocks
Calculations On Updated Blocks Only: [Disabled]
Clear Update Status After Full Calculations: [Enabled]
Calculator Cache With Multiple Bitmaps For: [Market]
```

```
[Thu Apr 6 10:09:21 1995] Local/Sample/Basic/Qatest/Info(1012672)
Calculator Information Message: Completion Notice For Block Number [49]
```

```
[Thu Apr 6 10:09:22 1995] Local/Sample/Basic/Qatest/Info(1012672)
Calculator Information Message: Completion Notice For Block Number [97]
```

```
[Thu Apr 6 10:09:24 1995] Local/Sample/Basic/Qatest/Info(1012672)
Calculator Information Message: Completion Notice For Block Number [145]
```

```
[Thu Apr 6 10:09:25 1995] Local/Sample/Basic/Qatest/Info(1012672)
Calculator Information Message: Completion Notice For Block Number [193]
```

```
[Thu Apr 6 10:09:27 1995] Local/Sample/Basic/Qatest/Info(1012672)
Calculator Information Message: Completion Notice For Block Number [241]
```

See Also

- [the section called “CALCNOTICE”](#)
- [SET MSG](#)
- [SET Commands](#)

SET REMOTECALC

For applications with transparent partitions, turns remote calculation to the source on or off.

Syntax

```
SET REMOTECALC ON | OFF;
```

Parameter Description

ON	Default. Essbase connects to the source partition enabling remote calculations.
OFF	Essbase does not connect to the source partition. Use this option only when absolutely sure the calculation script does not involve access to remote data.

Notes

- When you are working with transparent partitions and are sure that a calculation script does not include remote values in the calculations, you can use SET REMOTECALC OFF to improve calculation performance.
- Performance improvement is visible only when batch calculation is run on the target application.

Example

```
SET REMOTECALC ON;
```

```
SET REMOTECALC OFF;
```

See Also

- [SET Commands](#)

SET SCAPERSPECTIVE

Sets the perspective for varying attribute calculations.

Syntax

```
SET SCAPERSPECTIVE (mbrName1) [, (mbrName2)] ... [, (mbrNamen)] on Attribute_Dimension  
| OFF ;
```

Parameter	Description
<i>mbrName1</i> [,...] on <i>Attribute_Dimension</i>	Any valid single member name, or list of member names, on the specified varying attribute dimension.
OFF	Turn off the perspective setting for the calculation block.

Notes

- For use only in applications enabled with varying attributes.
- Only one independent member from each independent dimension is supported.

Example

Once the perspective is specified using this command, [@WITHATTR](#) can be used on a varying attribute inside a FIX statement. In the following example, the SET SCAPERSPECTIVE statements indicate that for attribute dimensions TYPE and TITLE, the subsequent FIX statement with [@WithATTR](#) will use their attribute association as defined at time FY03 and Jan.

```
set SCAPerspective ((FY03), (Jan)) on TYPE;  
set SCAPerspective ((FY03), (Jan)) on TITLE;
```

```
FIX (@WithAttr (TYPE, "=", Contractor), @withattr (Title, "=", Senior_QA_Engineer),  
Local, "HSP_Historical", "BU Version_1", Target, Local, FY03)  
HSP_INPUTVALUE = 100;  
ENDFIX;
```

See Also

- [@ISATTRIBUTE](#)
- [@ISMBRWITHATTR](#)
- [@WITHATTR](#)

SET UPDATECALC

Turns Intelligent Calculation on or off.

Syntax

```
SET UPDATECALC ON | OFF;
```

Parameter Description

ON Essbase calculates only blocks marked as dirty (see Description). Dirty blocks include updated blocks and their dependent parents (see Notes). The default setting is ON. You can change this default using the UPDATECALC TRUE | FALSE setting in the `essbase.cfg` file.

OFF Essbase calculates all data blocks, regardless of whether they have been updated.

Notes

- Using Intelligent Calculation, Essbase calculates only dirty blocks, such as updated data blocks and their dependent parents. Therefore, the calculation is very efficient.
- All data blocks in the database are marked as either clean or dirty. If a data block is clean, then Essbase knows that the block does not need to be recalculated.
- By default, all data blocks are marked as clean after a full calculation of the database but not after a partial calculation of the database. If required, you can change this default behavior using the SET CLEARUPDATESTATUS command in your calculation script.

- There are several possible reasons blocks might be marked as dirty. See the *Oracle Essbase Database Administrator's Guide* for information on Intelligent Calculation and clean and dirty blocks.

Example

```
SET UPDATECALC ON;
```

```
SET UPDATECALC OFF;
```

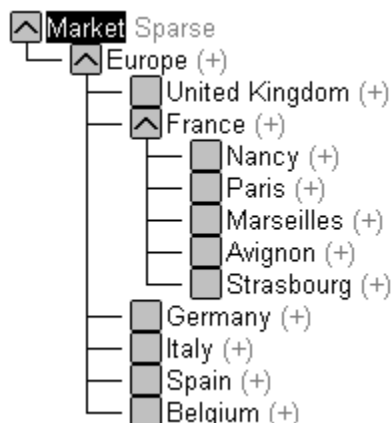
See Also

- [SET CLEARUPDATESTATUS](#)
- [“UPDATECALC” on page 513](#)
- [SET Commands](#)

SET UPTOLOCAL

Restricts consolidations to those parents with the same defined currency. The default is OFF.

For example, all cities in Switzerland use the Swiss franc (CHF) as the unit of currency. Therefore, all children of Switzerland, such as the cities Geneva, Zurich, and Lucerne, consolidate to Switzerland. Consolidation stops at this level, however, because Europe also contains countries that use other currencies. The following database outline example illustrates this situation:



If you want to consolidate values above this level, you must use [CCONV](#) to convert the values to a master rate before consolidating.

Syntax

```
SET UPTOLOCAL ON | OFF ;
```

Notes

SET UPTOLOCAL ON has no effect on databases with no currency definitions.

Example

```
SET UPTOLOCAL ON;
```

```
SET UPTOLOCAL OFF;
```

See Also

- [CCONV](#)
- [SET CCTRACKCALC](#)
- [CLEARCCTRACK](#)
- [“CCTRACK” on page 408](#)

VAR

Declares a temporary variable that contains a single value.

Note: You can also use a single VAR command to declare multiple variables by supplying a comma-delimited list of variable names.

Syntax

```
VAR varName [= value] ;
```

Parameter Description

varName Name of the temporary variable.

value Optional parameter that declares the data value.

Notes

- The name of the variable cannot duplicate a database member name.
- If a value is not declared, it is set to #MISSING.
- VAR commands can only be assigned values within a member calculation or when VAR is declared.

Example

```
VAR Target = 1200;
```

```
VAR Break1, Break2, Break3;
```

See Also

- [ARRAY](#)

4

Essbase.cfg Configuration Settings

In This Chapter

Configuration File Overview	369
Configuring Essbase.cfg	369
Essbase.cfg Setting Categorical List	370
Aggregate Storage and Block Storage Settings Comparison	376
Configuration Settings Reference	380

Configuration File Overview

With the `essbase.cfg` configuration file, you can customize your Essbase Server configuration. Settings specified in the `essbase.cfg` file usually apply to the entire Essbase Server. These settings override the Essbase defaults and apply to all databases within all applications on the Essbase Server.

You can create one `essbase.cfg` file for server settings, and another for client settings. Assume settings are for the server unless otherwise noted.

Configuring Essbase.cfg

A default `essbase.cfg` file exists in the Essbase `bin` directory.

➤ To edit the `essbase.cfg` configuration file:

- 1 **Open the file with a text editor.**
- 2 **Enter each setting on a separate line in the file. Semicolon terminators are not required.**
- 3 **Save the file as `essbase.cfg` in the `bin` directory.**
- 4 **After editing the configuration file, perform the proper action to have the configuration file reread:**
 - If the setting applies to the server, stop and restart Essbase Server.
 - If the setting applies to a specific application, stop the application (if it is running) and restart it.
 - If the setting applies only to a database, restart the application.

Notes

- Oracle recommends that you make there are no duplicate settings in the `essbase.cfg` file.
- You can override many `essbase.cfg` values using:
 - MaxL statements
 - Administration Services dialogs
 - ESSCMD commands
- When you use MaxL or Administration Services to change `essbase.cfg` values, many values are effective immediately. See the *Oracle Essbase Database Administrator's Guide* for details.
- Some `essbase.cfg` settings affect performance. Before you override Essbase defaults, see information about performance optimization and storage settings in the *Oracle Essbase Database Administrator's Guide*
- `essbase.cfg` settings apply to all databases unless the values are noted as database- or application-specific.
- Essbase uses the keywords and their unparsed values "as is." No syntax check is performed.
- You can use an `essbase.cfg` file on the client to override Essbase default network settings. Only the following settings can be used in an `essbase.cfg` client file:
 - [“AGENTPORT” on page 387](#)
 - [“APSRESOLVER” on page 392](#)
 - [“NETDELAY” on page 473](#)
 - [“NETRETRYCOUNT” on page 474](#)
 - [“PORTINC” on page 481](#)
 - [“SERVERPORTBEGIN” on page 493](#)
 - [“SERVERPORTEND” on page 494](#)

All other configuration settings are intended for the server `essbase.cfg` file only.

Example

The following is an example of `essbase.cfg` server file entries:

```
SSPROCROWLIMIT 20000  
LOCKTIMEOUT 1200
```

See the *Oracle Essbase Database Administrator's Guide*.

Essbase.cfg Setting Categorical List

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

- [Backup and Recovery Configuration Settings](#)

- Calculation Configuration Settings
- Data Import and Export Configuration Settings
- Hybrid Analysis Configuration Settings
- Failover Clustering 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
- “Security File Configuration Settings” on page 376

Backup and Recovery Configuration Settings

- “TRANSACTIONLOGDATALOADARCHIVE” on page 508
- “TRANSACTIONLOGLOCATION” on page 510

Calculation Configuration Settings

- “AGGRESSIVEBLKOPTIMIZATION” on page 389
- “CALCCACHE” on page 396
- “CALCCACHEHIGH” on page 397
- “CALCCACHEDEFAULT” on page 398
- “CALCCACHELOW” on page 399
- “CALCLIMITFORMULARECURSION” on page 400
- “CALCLOCKBLOCK” on page 401
- “CALCMODE” on page 402
- “CALCNOTICE” on page 403
- “CALCOPTFRMLBOTTOMUP” on page 404
- “CALCPARALLEL” on page 406
- “CALCREUSEDYNCALCBLOCKS” on page 405
- “CALCTASKDIMS” on page 407
- “CCTRACK” on page 408
- “DYNCALCCACHEBLKRELEASE” on page 429
- “DYNCALCCACHEBLKTIMEOUT” on page 430
- “DYNCALCCACHECOMPRBLKBUFSIZE” on page 432

- “DYNCALCCACHEMAXSIZE” on page 433
- “DYNCALCCACHEONLY” on page 435
- “DYNCALCCACHEWAITFORBLK” on page 436
- “EXCLUSIVECALC” on page 444
- “FORCEALLDENSECALCON2PASSACCOUNTS” on page 446
- “MULTIPLEBITMAPMEMCHECK” on page 472
- “PARCALCMULTIPLEBITMAPMEMOPT” on page 479
- “RTDEPCALCOPTIMIZE” on page 490
- “UPDATECALC” on page 513

Data Import and Export Configuration Settings

- “DATAEXPORTENABLEBATCHINSERT” on page 414
- “DEXPSQLROWSIZE” on page 418
- “DLSINGLETHREADPERSTAGE” on page 424
- “DLTHREADSPREPARE” on page 426
- “DLTHREADSWRITE” on page 427
- “EXPORTTHREADS” on page 444

Hybrid Analysis Configuration Settings

- “HAENABLE” on page 449
- “HAMAXNUMCONNECTION” on page 449
- “HAMAXNUMSQLQUERY” on page 450
- “HAMAXQUERYROWS” on page 451
- “HAMAXQUERYTIME” on page 452
- “HAMEMORYCACHESIZE” on page 453
- “HARAGGEDHIERARCHY” on page 454
- “HARETRIEVENUMROW” on page 455
- “HASOURCEDSNOS390” on page 456

Failover Clustering Configuration Settings

- “AGENTLEASEEXPIRATIONTIME” on page 384
- “AGENTLEASEMAXRETRYCOUNT” on page 385
- “AGENTLEASERENEWALTIME” on page 385
- “APSRESOLVER” on page 392

- “FAILOVERMODE” on page 445
- “ESSBASEFAILOVERTRACELEVEL” on page 441
- “SERVERLEASEEXPIRATIONTIME” on page 492
- “SERVERLEASEMAXRETRYCOUNT” on page 492
- “SERVERLEASERENEWALTIME” on page 493

Logging and Error Handling Configuration Settings

- “AGENTDISPLAYMESSAGELEVEL” on page 383
- “AGENTLOGMESSAGELEVEL” on page 386
- “AGTMAXLOGFILESIZE” on page 390
- “APPMAXLOGFILESIZE” on page 392
- “CALCNOTICE” on page 403
- “CLEARLOGFILE” on page 410
- “CRASHDUMP” on page 411
- “DATAERRORLIMIT” on page 413
- “DELIMITEDMSG” on page 417
- “DELIMITER” on page 417
- “DISPLAYMESSAGELEVEL” on page 423
- “EXCEPTIONLOGOVERWRITE” on page 442
- “ESSBASEFAILOVERTRACELEVEL” on page 441
- “GRIDEXPANSIONMESSAGES” on page 448
- “IBHFIXTHRESHOLD” on page 457
- “LOGINFAILUREMESSAGEDETAILED” on page 465
- “LOGMESSAGELEVEL” on page 465
- “NOMSGLOGGINGONDATAERRORLIMIT” on page 476
- “OUTLINECHANGELOG” on page 478
- “OUTLINECHANGELOGFILESIZE” on page 479
- “SILENTOTLQUERY” on page 497
- “SQLFETCHERRORPOPUP” on page 498
- “TIMINGMESSAGES” on page 507
- “SSINVALIDTEXTDETECTION” on page 501
- “UNICODEAGENTLOG” on page 512

Memory Management Configuration Settings

- “ASOLOADBUFFERWAIT” on page 393
- “DATACACHESIZE” on page 412
- “DATAFILECACHESIZE” on page 415
- “DYNCALCCACHEMAXSIZE” on page 433
- “HAMEMORYCACHESIZE” on page 453
- “INDEXCACHESIZE” on page 463
- “MEMSCALINGFACTOR” on page 471
- “MAXFORMULACACHESIZE” on page 467
- “MULTIPLEBITMAPMEMCHECK” on page 472
- “PARCALCMULTIPLEBITMAPMEMOPT” on page 479
- “PRELOADALIASNAMESPACE” on page 483
- “PRELOADMEMBERNAMESPACE” on page 483
- “PRELOADUDANAMESPACE” on page 484
- “SSOPTIMIZEDGRIDPROCESSING” on page 503
- “SSPROCROWLIMIT” on page 504
- “TRIGMAXMEMSIZE” on page 511
- “VLBREPORT” on page 514

Miscellaneous Configuration Settings

- “AUTHENTICATIONMODULE” on page 395
- “DELAYEDRECOVERY” on page 416
- “DIRECTIO” on page 420
- “DISBLEREPLMISSINGDATA” on page 421
- “DISKVOLUMES” on page 422
- “INCRESTRUC” on page 460
- “JVMMODULELOCATION” on page 463
- “LROONSHAREDMBR” on page 466
- “NUMERICPRECISION” on page 477
- “TARGETTIMESERIESOPT” on page 507

Partitioning Configuration Settings

- “ENABLE_DIAG_TRANSPARENT_PARTITION” on page 438
- “MAX_REQUEST_GRID_SIZE” on page 468

- [“MAX_RESPONSE_GRID_SIZE” on page 469](#)
- [“REPLICATIONASSUMEIDENTICALOUTLINE” on page 489](#)

Ports and Connections Configuration Settings

- [“AGENTDELAY” on page 382](#)
- [“AGENTDESC” on page 383](#)
- [“AGENTPORT” on page 387](#)
- [“AGENTTHREADS” on page 388](#)
- [“AGTSVRCONNECTIONS” on page 391](#)
- [“APSRESOLVER” on page 392](#)
- [“MAXLOGINS” on page 468](#)
- [“NETBINDRETRYDELAY” on page 473](#)
- [“NETDELAY” on page 473](#)
- [“NETRETRYCOUNT” on page 474](#)
- [“NETTCPCONNECTRETRYCOUNT” on page 475](#)
- [“PIPEBUFFERSIZE” on page 480](#)
- [“PORTINC” on page 481](#)
- [“PORTUSAGELOGINTERVAL” on page 482](#)
- [“SERVERPORTBEGIN” on page 493](#)
- [“SERVERPORTEND” on page 494](#)
- [“SERVERTHREADS” on page 496](#)

See Also

[“SSL Configuration Settings” on page 376](#)

Query Management Configuration Settings

- [“FORCEGRIDEXPANSION” on page 447](#)
- [“GRIDEXPANSION” on page 448](#)
- [“GRIDEXPANSIONMESSAGES” on page 448](#)
- [“HAMAXNUMSQLQUERY” on page 450](#)
- [“HAMAXQUERYROWS” on page 451](#)
- [“HAMAXQUERYTIME” on page 452](#)
- [“LOCKTIMEOUT” on page 464](#)
- [“PRELOADUDANAMESPACE” on page 484](#)
- [“QRYGOVEXECBLK” on page 485](#)

- “QRYGOVEXECTIME” on page 486
- “SSAUDIT” on page 499
- “SSAUDITR” on page 500
- “SSLCIPHERSUITES” on page 502
- “SSLOGUNKNOWN” on page 503
- “SSOPTIMIZEDGRIDPROCESSING” on page 503
- “SSPROCROWLIMIT” on page 504
- “SUPNA” on page 505
- “TARGETASOOPT” on page 506
- “WALLETPATH” on page 515

See also Chapter 8, “Query Logging Configuration,” which you can enable by means of a separate configuration file.

Security File Configuration Settings

- “ENABLESWITCHTOBACKUPFILE” on page 440
- “NUMBEROFSECFILEBACKUPS” on page 476
- “SECFILEBACKUPINTERVAL” on page 490
- “SECURITYFILECOMPACTIONPERCENT” on page 491

SSL Configuration Settings

- “AGENTSECUREPORT” on page 388
- “CLIENTPREFERREDMODE” on page 411
- “ENABLECLEARMODE” on page 439
- “ENABLESECUREMODE” on page 440
- “NETSSLHANDSHAKETIMEOUT” on page 474
- “SSLCIPHERSUITES” on page 502
- “WALLETPATH” on page 515

See the *Oracle Hyperion Enterprise Performance Management System Security Administration Guide*.

Aggregate Storage and Block Storage Settings Comparison

The following settings apply only to aggregate storage databases.

- “ASOLOADBUFFERWAIT” on page 393
- “ASOSAMPLESIZEPERCENT” on page 394
- “MAX_REQUEST_GRID_SIZE” on page 468
- “MAX_RESPONSE_GRID_SIZE” on page 469
- “PRELOADALIASNAMESPACE” on page 483
- “PRELOADMEMBERNAMESPACE” on page 483
- “REPLICATIONASSUMEIDENTICALOUTLINE” on page 489

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

“AGENTDELAY” on page 382

“AGENTDISPLAYMESSAGELEVEL” on page 383

“AGENTLEASEEXPIRATIONTIME” on page 384

“AGENTLEASEMAXRETRYCOUNT” on page 385

“AGENTLEASERENEWALTIME” on page 385

“AGENTLOGMESSAGELEVEL” on page 386

“AGENTPORT” on page 387

“AGENTTHREADS” on page 388

“AGTSVRCONNECTIONS” on page 391

“AUTHENTICATIONMODULE” on page 395

“CALCLIMITFORMULARECURSION” on page 400

“CALCPARALLEL” on page 406

“CLEARLOGFILE” on page 410

“CRASHDUMP” on page 411

“DATAERRORLIMIT” on page 413

“DELIMITEDMSG” on page 417

“DELIMITER” on page 417

“DISPLAYMESSAGELEVEL” on page 423

“DLSINGLETHREADPERSTAGE” on page 424

“DLTHREADSPREPARE” on page 426

“ENABLE_DIAG_TRANSPARENT_PARTITION” on page 438

“ENABLESWITCHTOBACKUPFILE” on page 440

“EXCEPTIONLOGOVERWRITE” on page 442

“FAILOVERMODE” on page 445

“FORCEGRIDEXPANSION” on page 447

“GRIDEXPANSION” on page 448

“GRIDEXPANSIONMESSAGES” on page 448

“HAENABLE” on page 449

“HAMAXNUMCONNECTION” on page 449

“HAMAXNUMSQLQUERY” on page 450

“HAMAXQUERYROWS” on page 451

“HAMAXQUERYTIME” on page 452

“HAMEMORYCACHESIZE” on page 453

“HARAGGEDHIERARCHY” on page 454

“HARETRIEVENUMROW” on page 455

“HASOURCEDSNOS390” on page 456

“JVMMODULELOCATION” on page 463

“LOGINFAILUREMESSAGEDETAILED” on page 465

“LOGMESSAGELEVEL” on page 465

“MAXLOGINS” on page 468

“NETBINDRETRYDELAY” on page 473

“NETDELAY” on page 473

“NETRETRYCOUNT” on page 474

“NOMSGLOGGINGONDATAERRORLIMIT” on page 476

“NUMERICPRECISION” on page 477

“NUMBEROFSECFILEBACKUPS” on page 476

“OUTLINECHANGELOG” on page 478

“OUTLINECHANGELOGFILESIZE” on page 479

“PIPEBUFFERSIZE” on page 480

“PORTINC” on page 481

“PORTUSAGELOGINTERVAL” on page 482

“PRELOADUDANAMESPACE” on page 484

“QRYGOVEXEETIME” on page 486

“SECFILEBACKUPINTERVAL” on page 490

“SECURITYFILECOMPACTIONPERCENT” on page 491

“SERVERLEASEEXPIRATIONTIME” on page 492

“SERVERLEASEMAXRETRYCOUNT” on page 492

“SERVERLEASERENEWALTIME” on page 493

“SERVERPORTBEGIN” on page 493
“SERVERPORTEND” on page 494
“SERVERTHREADS” on page 496
“SILENTOTLQUERY” on page 497
“SQLFETCHERRORPOPUP” on page 498
“SSAUDITR” on page 500
“SSLOGUNKNOWN” on page 503
“SSOPTIMIZEDGRIDPROCESSING” on page 503
“SSPROCROWLIMIT” on page 504
“SUPNA” on page 505
“TARGETTIMESERIESOPT” on page 507
“TIMINGMESSAGES” on page 507
“TRANSACTIONLOGDATALOADARCHIVE” on page 508
“TRANSACTIONLOGLOCATION” on page 510
“TRIGMAXMEMSIZE” on page 511
“UNICODEAGENTLOG” on page 512

The following settings apply only to block storage databases.

“AGGRESSIVEBLKOPTIMIZATION” on page 389
“CALCCACHE” on page 396
“CALCCACHEHIGH” on page 397
“CALCCACHEDEFAULT” on page 398
“CALCCACHELOW” on page 399
“CALCLIMITFORMULARECURSION” on page 400
“CALCLOCKBLOCK” on page 401
“CALCMODE” on page 402
“CALCNOTICE” on page 403
“CALCOPTFRMLBOTTOMUP” on page 404
“CALCREUSEDYNALCBLOCKS” on page 405
“CALCTASKDIMS” on page 407
“CCTRACK” on page 408
“DATACACHESIZE” on page 412
“DATAEXPORTENABLEBATCHINSERT” on page 414
“DATAFILECACHESIZE” on page 415

“DELAYEDRECOVERY” on page 416
“DEXPSQLROWSIZE” on page 418
“DIRECTIO” on page 420
“DISKVOLUMES” on page 422
“DLTHREADSWRITE” on page 427
“DYNALCCACHEBLKRELEASE” on page 429
“DYNALCCACHEBLKTIMEOUT” on page 430
“DYNALCCACHECOMPRBLKBUFSIZE” on page 432
“DYNALCCACHEMAXSIZE” on page 433
“DYNALCCACHEONLY” on page 435
“DYNALCCACHEWAITFORBLK” on page 436
“EXCLUSIVECALC” on page 444
“FORCEALLDENSECALCON2PASSACCOUNTS” on page 446
“EXPORTTHREADS” on page 444
“IBHFIXTHRESHOLD” on page 457
“INCRESTRUC” on page 460
“INDEXCACHESIZE” on page 463
“LOCKTIMEOUT” on page 464
“LROONSHAREDMBR” on page 466
“MULTIPLEBITMAPMEMCHECK” on page 472
“PARCALCMULTIPLEBITMAPMEMOPT” on page 479
“QRYGOVEXECBLK” on page 485
“SSAUDIT” on page 499
“UPDATECALC” on page 513
“VLBREPORT” on page 514

Configuration Settings Reference

Consult the Contents pane for a categorical list of configuration settings.

AGENTDELAY	DYNALCCACHEBLKTIMEOUT	NOMSGLOGGINGONDATAERRORLIMIT
AGENTDESC	DYNALCCACHECOMPRBLKBUFSIZE	NUMBEROFSECFILEBACKUPS
AGENTDISPLAYMESSAGELEVEL	DYNALCCACHEMAXSIZE	NUMERICPRECISION
AGENTLEASEEXPIRATIONTIME	DYNALCCACHEONLY	OUTLINECHANGELOG

AGENTDELAY	DYNCALCCACHEBLKTIMEOUT	NOMSGLOGGINGONDATAERRORLIMIT
AGENTLEASEMAXRETRYCOUNT	DYNCALCCACHEWAITFORBLK	OUTLINECHANGELOGFILESIZE
AGENTLEASERENEWALTIME	ENABLE_DIAG_TRANSPARENT_PARTITION	PARCALCMULTIPLEBITMAPMEMOPT
AGENTLOGMESSAGELEVEL	ENABLECLEARMODE	PERSISTUSERATLOGIN
AGENTPORT	ENABLESECUREMODE	PIPEBUFFERSIZE
AGENTSECUREPORT	ENABLESWITCHTOBACKUPFILE	PORTINC
AGENTTHREADS	ESSBASEFAILOVERTRACELEVEL	PORTUSAGELOGINTERVAL
AGGRESSIVEBLKOPTIMIZATION	ESSBASESERVERHOSTNAME	PRELOADALIASNAMESPACE
AGTMAXLOGFILESIZE	EXCEPTIONLOGOVERWRITE	PRELOADMEMBERNAMESPACE
AGTSVRCONNECTIONS	EXCLUSIVECALC	PRELOADUDANAMESPACE
APPMAXLOGFILESIZE	EXPORTTHREADS	QRYGOVEXECLBK
APSRESOLVER	FAILOVERMODE	QRYGOVEXECTIME
ASOLOADBUFFERWAIT	FILELOCKINGMODE	REPLAYSECURITYOPTION
ASOSAMPLESIZEPERCENT	FORCEALLDENSECALCON2PASSACCOUNTS	REPLICATIONASSUMEIDENTICALOUTLI
AUTHENTICATIONMODULE	FORCEGRIDEXPANSION	RTDEPCALCOPTIMIZE
CALCCACHE	GRIDEXPANSION	SECFILEBACKUPINTERVAL
CALCCACHEHIGH	GRIDEXPANSIONMESSAGES	SECURITYFILECOMPACTIONPERCENT
CALCCACHEDEFAULT	HAENABLE	SERVERLEASEEXPIRATIONTIME
CALCCACHELOW	HAMAXNUMCONNECTION	SERVERLEASEMAXRETRYCOUNT
CALCLIMITFORMULARECURSION	HAMAXNUMSQLQUERY	SERVERLEASERENEWALTIME
CALCLOCKBLOCK	HAMAXQUERYROWS	SERVERPORTBEGIN
CALCMODE	HAMAXQUERYTIME	SERVERPORTEND
CALCNOTICE	HAMEMORYCACHESIZE	SERVERTHREADS
CALCOPTFRMLBOTTOMUP	HARAGGEDHIERARCHY	SILENTOTLQUERY
CALCREUSEDYNCALCBLOCKS	HARETRIEVENUMROW	SPLITARCHIVEFILE
CALCPARALLEL	HASOURCEDSNOS390	SQLFETCHERRORPOPUP
CALCTASKDIMS	HISLEVELDRILLTHROUGH	SSAUDIT
CCTRACK	IBHFIXTHRESHOLD	SSAUDITR
CLEARLOGFILE	IDMIGRATION	SSINVALIDTEXTDETECTION
CLIENTPREFERREDMODE	IMPLIED_SHARE	SSLCIPHERSUITES

AGENTDELAY	DYNCALCCACHEBLKTIMEOUT	NOMSGLOGGINGONDATAERRORLIMIT
CRASHDUMP	INCRESTRUC	SSLOGUNKNOWN
DATACACHE SIZE	INDEXCACHE SIZE	SSOPTIMIZEDGRIDPROCESSING
DATAERRORLIMIT	JVMMODULELOCATION	SSPROCROWLIMIT
DATAEXPORTENABLEBATCHINSERT	LOCKTIMEOUT	SUPNA
DATAFILECACHE SIZE	LOGINFAILUREMESSAGEDETAILED	TARGETASOOPT
DEFAULTLOGLOCATION	LOGMESSAGELEVEL	TARGETTIMESERIESOPT
DELAYEDRECOVERY	LROONSHAREDMBR	TIMINGMESSAGES
DELIMITEDMSG	MAXERRORMBRVERIFYREPORT	TRANSACTIONLOGDATALOADARCHIVE
DELIMITER	MAXFORMULACACHE SIZE	TRANSACTIONLOGLOCATION
DEXPSQLROWSIZE	MAXLOGINS	TRIGMAXMEMSIZE
DIMBUILDERRORLIMIT	MAX_REQUEST_GRID_SIZE	UNICODEAGENTLOG
DIMBUILDSTATSINTERVAL	MAX_RESPONSE_GRID_SIZE	UPDATECALC
DIRECTIO	MDXFORMULARECURSIONLIMIT	VLBREPORT
DISABLEREPLMISSINGDATA	MEMSCALINGFACTOR	WALLETPATH
DISKVOLUMES	MULTIPLEBITMAPMEMCHECK	XOLAPENABLEHEURISTICS
DISPLAYMESSAGELEVEL	NETBINDRETRYDELAY	XOLAPMAXNUMCONNECTION
DLSINGLETHREADPERSTAGE	NETDELAY	XOLAPSCHEMAVERIFICATION
DLTHREADSPREPARE	NETRETRYCOUNT	XOLAPSQLIDLEPERIOD
DLTHREADSWRITE	NETSSLHANDSHAKETIMEOUT	
DYNCALCCACHEBLKRELEASE	NETTCPCONNECTRETRYCOUNT	

AGENTDELAY

Specifies the number of seconds an Agent thread waits to perform a specific action.

Syntax

AGENTDELAY *n*

Where *n* is the number of seconds an Agent thread waits before performing a specific action. *n* must be an integer and must be 5 or higher. The default value is 20.

Description

AGENTDELAY specifies the number of seconds an Agent thread waits for a resource to become available so it can perform a specific action. If the resource is still unavailable when the specified value for AGENTDELAY is completely used, the agent times out and does not complete the transaction.

Notes

The higher the value of AGENTTHREADS, the more contention for resources there is, and therefore the higher the AGENTDELAY value needs to be.

Example

```
AGENTDELAY 60
```

See Also

[“AGENTTHREADS” on page 388](#)

[“AGTSVRCONNECTIONS” on page 391](#)

AGENTDESC

When the Configuration Utility is used to register an Essbase Server Agent as a Windows service, the text entered in the Service Name Identifier field is stored as AGENTDESC in the Essbase configuration file (essbase.cfg).

Syntax

```
AGENTDESC description
```

Where *description* is the unique description provided for an Essbase Agent Windows service when it was registered through the Configuration Utility.

See Also

[“AGENTPORT” on page 387](#)

[“SERVERPORTBEGIN” on page 493](#)

[“SERVERPORTEND” on page 494](#)

[“PORTINC” on page 481](#)

AGENTDISPLAYMESSAGELEVEL

Sets the message types that will be displayed in the Essbase Server console. Only the console is affected. To set the level of messages written to the Essbase Server *log*, use AGENTLOGMESSAGELEVEL.

Syntax

```
AGENTDISPLAYMESSAGELEVEL INFO | WARNING | ERROR
```

Where INFO, WARNING, ERROR are levels:

- INFO—When specified, all three types of messages are displayed in the Essbase Server console. This is the default.
- WARNING—When specified, only Warning and Error messages are displayed in the Essbase Server console.
- ERROR—When specified, only error messages are displayed in the Essbase Server console. No Warning or Info messages are displayed.

Description

AGENTDISPLAYMESSAGELEVEL enables the level of messages displayed in the Essbase Server console to be specified.

Notes

This setting affects only the messages displayed in the Essbase Server console. To control the messages written to the Essbase Server log, use [“AGENTLOGMESSAGELEVEL” on page 386](#). To set the same level for both the console and the log, use both settings.

Example

```
AGENTDISPLAYMESSAGELEVEL WARNING
```

Sets the message level at Warning. Only Warning and Error messages are displayed in the Essbase Server console.

See Also

[SETMSGLEVEL](#)

[“AGENTLOGMESSAGELEVEL” on page 386](#)

AGENTLEASEEXPIRATIONTIME

Sets the maximum amount of time that Essbase Agent can own a lease before the lease is terminated.

Syntax

```
AGENTLEASEEXPIRATIONTIME n
```

Where *n* is an integer specifying the number of seconds before a lease expires. The default value is 20.

Example

```
AGENTLEASEEXPIRATIONTIME 20
```


See Also

[“AGENTLEASEMAXRETRYCOUNT” on page 385](#)

[“AGENTLEASERENEWALTIME” on page 385](#)

[“SERVERLEASEEXPIRATIONTIME” on page 492](#)

[“SERVERLEASEMAXRETRYCOUNT” on page 492](#)

[“SERVERLEASERENEWALTIME” on page 493](#)

AGENTLEASEMAXRETRYCOUNT

Specifies the number of times that Essbase Agent attempts to acquire or renew a lease. If the attempts are unsuccessful, the agent terminates itself.

Syntax

```
AGENTLEASEMAXRETRYCOUNT n
```

Where *n* is an integer. The default value is 5.

Example

```
AGENTLEASEMAXRETRYCOUNT 5
```

See Also

[“AGENTLEASEEXPIRATIONTIME” on page 384](#)

[“AGENTLEASERENEWALTIME” on page 385](#)

[“SERVERLEASEEXPIRATIONTIME” on page 492](#)

[“SERVERLEASEMAXRETRYCOUNT” on page 492](#)

[“SERVERLEASERENEWALTIME” on page 493](#)

AGENTLEASERENEWALTIME

Specifies the time interval, in seconds, after which Essbase Agent attempts to renew a lease. This value must be less than the value of AGENTLEASEEXPIRATIONTIME.

Syntax

```
AGENTLEASERENEWALTIME n
```

Where *n* is an integer specifying the number of seconds to reestablish ownership after a lease expires. The default value is 10.

Example

```
AGENTLEASERENEWALTIME 10
```

See Also

[“AGENTLEASEEXPIRATIONTIME” on page 384](#)

[“AGENTLEASEMAXRETRYCOUNT” on page 385](#)

[“SERVERLEASEEXPIRATIONTIME” on page 492](#)

[“SERVERLEASEMAXRETRYCOUNT” on page 492](#)

[“SERVERLEASERENEWALTIME” on page 493](#)

AGENTLOGMESSAGELEVEL

Sets the message types that will be written to the Essbase Server log, `essbase.log`.

Syntax

```
AGENTLOGMESSAGELEVEL INFO | WARNING | ERROR | DEBUG
```

Where INFO, WARNING, ERROR, and DEBUG are levels:

- INFO—Info, Warning, and Error messages are written to the Essbase Server log. This is the default setting.
- WARNING—Only Warning and Error messages are written to the Essbase Server log.
- ERROR—Only Error messages are written to the Essbase Server log. No Warning or Info messages are written to the .
- DEBUG—OPMN ping messages (Received OPMN Ping Request and Sent the Response to OPMN Ping) are included in the Essbase Server log.

Description

AGENTLOGMESSAGELEVEL enables the level of messages written to the Essbase Server log to be specified.

Notes

To control the messages displayed in the Agent console, use [“AGENTDISPLAYMESSAGELEVEL” on page 383](#). To set the same level for both the console and the log, use both settings.

Example

```
AGENTLOGMESSAGELEVEL WARNING
```

Sets the message level at Warning. Only Warning and Error messages are written to the Essbase Server log.

See Also

[SETMSGLEVEL](#)

[“AGENTDISPLAYMESSAGELEVEL” on page 383](#)

AGENTPORT

Specifies the port that the Agent uses.

Syntax

```
AGENTPORT n
```

Where *n* is the port number for the Agent. This port number should not be in use by any other process. The default value is 1423.

Description

AGENTPORT specifies the port that the Agent uses.

You may wish to change the default for many reasons. These are two common reasons:

- The first server port, 1423, is inappropriate for your site.
- You may wish to install a second Agent on a single computer to facilitate testing. Use AGENTPORT and the related configuration settings to assign the second Agent to a different port than the first. Use AGENTPORT with SERVERPORTBEGIN, SERVERPORTEND, and PORTINC.

Caution! Do not use more than one Agent per computer in production systems.

Notes

- The setting is needed only in the server configuration file.
- You must perform other steps to enable multiple agents on one computer. Please see the *Oracle Essbase Database Administrator's Guide* for instructions.

Example

```
AGENTPORT 1478
SERVERPORTBEGIN 32470
SERVERPORTEND 32600
PORTINC 5
```

This example produces these results:

- AGENTPORT sets the port that the Agent will use at 1478.
- SERVERPORTBEGIN sets the value that the first server process will try to use for a port at 32470.
- SERVERPORTEND sets the highest port number value this installation can use.
- PORTINC controls the increment value used for each port. In this example, if the first server process used port number 32470, then the next process would use 32475.

See Also

[“SERVERPORTBEGIN” on page 493](#)

[“SERVERPORTEND” on page 494](#)

[“PORTINC” on page 481](#)

[“PORTUSAGELOGINTERVAL” on page 482](#)

AGENTSECUREPORT

Specifies the port that the agent uses for secure communication using Secure Socket Layer (SSL).

Syntax

```
AGENTSECUREPORT n
```

Where *n* is the port number for the agent. This port number should not be in use by any other process. The default value is 6423.

Description

AGENTSECUREPORT specifies the port that the agent uses for secure communication using SSL.

Example

```
AGENTSECUREPORT 16001
```

See Also

[“CLIENTPREFERREDMODE” on page 411](#)

[“ENABLECLEARMODE” on page 439](#)

[“ENABLESECUREMODE” on page 440](#)

[“NETSSLHANDSHAKETIMEOUT” on page 474](#)

[“SSLCIPHERSUITES” on page 502](#)

[“WALLETPATH” on page 515](#)

For information on implementing SSL, see the *Oracle Hyperion Enterprise Performance Management System Security Administration Guide*.

AGENTTHREADS

Specifies how many threads the Agent may spawn.

Syntax

```
AGENTTHREADS n
```

Where *n* is the number of threads that the Agent can spawn:

- Between 2 and 500 on 32-bit platforms
- Between 2 and 1024 on 64-bit platforms

The default value is 5. It is strongly recommended that you use this default value if you are running Essbase on a 32-bit platform. See Notes below.

Description

AGENTTHREADS specifies how many threads the Agent may spawn. Some of these threads are used in conjunction with AGTSVRCONNECTIONS to allow initial login via the Agent and to establish the first connection to application and database, negotiated by the Agent and server. The rest of the threads are used for other Agent tasks unrelated to AGTSVRCONNECTIONS. Once connected, these threads are no longer used. Client requests use server threads whose maximum number is governed by SERVERTHREAD and by the number of licensed ports purchased.

When a connection is requested, the Agent assigns a thread to the request and releases the thread when the connection is made.

Notes

- While the actual maximum value you can set is 500 (or 1024 on 64-bit platforms), the maximum number of threads an operating system can handle might be much lower. It is strongly recommended that you use the default value.
- If you want to set this parameter to a value higher than the default (5), check with your system administrator, as higher values can significantly consume system resources.
- If you choose a number less than 2, 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.

Example

```
AGENTTHREADS 15
```

See Also

[“AGTSVRCONNECTIONS” on page 391](#)

[“AGENTDELAY” on page 382](#)

AGGRESSIVEBLKOPTIMIZATION

Improves batch calculation time for block storage outlines.

This setting does not apply to aggregate storage databases.

Syntax

```
AGGRESSIVEBLKOPTIMIZATION TRUE | FALSE
```

- TRUE—Essbase uses batch calculation on smaller kernel blocks. Use only if there is no formula dependency on dense Dynamic Calc members.

- FALSE—Essbase does not use batch calculation on smaller kernel blocks. The default value is FALSE.

Description

When there are dense Dynamic Calc members in the outline, a batch calculation with formulas uses blocks that contain data cells for all dense Dynamic Calc members. Setting `AggressiveBlkOptimization` to TRUE makes batch calculation work on kernel blocks (smaller blocks) directly, which may improve performance. Use this setting only if there is no formula dependency on dense Dynamic Calc members; otherwise, the calculation may produce incorrect results.

Example

```
AGGRESSIVEBLKOPTIMIZATION TRUE
```

Improves calculation performance for outlines in which there is no formula dependency on dense Dynamic Calc members.

AGTMAXLOGFILESIZE

Sets the maximum size of the Essbase Server log file.

Syntax

```
AGTMAXLOGFILESIZE n
```

Where *n* is the file size in bytes:

- Minimum file size is 1 MB (1048576 bytes). User-specified values less than the minimum are not recognized and are reset to 1 MB.
- Maximum file size is 2 GB (2147483647 bytes). User-specified values greater than the maximum are not recognized and are reset to 2 GB.
- If no value is specified, the default value of 2 GB (2147483647) is used.

Description

This parameter enables the user to specify the maximum size for the Essbase Server log file.

For the location of `essbase.log`, see the *Oracle Essbase Database Administrator's Guide*.

The current log file is always `essbase.log`. When maximum log file size is reached, the file is renamed `essbase.log.n` (for example, `essbase.log.0`, `essbase.log.1`, and so on), and a new `essbase.log` file is created.

Example

```
AGTMAXLOGFILESIZE 1500000
```

Sets the maximum Agent log file size to 1500000 bytes.

AGTSVRCONNECTIONS

Specifies the maximum number of the Essbase Server can spawn to allow the first connection to an application and database, negotiated between the Agent and server.

Syntax

```
AGTSVRCONNECTIONS n
```

Where *n* is the number of threads that Essbase Server can spawn:

- Default value is 5
- Minimum value is 1
- Maximum value is the value set for AGENTTHREADS

Description

AGTSVRCONNECTIONS specifies the maximum number of threads that the Essbase Server can create to connect to the Agent. Each connection uses one thread only while logging in and connecting to an application and database. Once connected, client requests are managed by threads whose maximum is controlled by SERVERTHREADS and by the number of licensed ports.

You may wish to adjust this value from the default value 5 if, for example, you are expecting a large number of users to login and select a single application in a short period of time.

The configuration parameter AGENTTHREADS controls the maximum number of threads the agent can create, so keep the value of AGTSVRCONNECTIONS equal to or less than the value of AGENTTHREADS to avoid wasting resources.

Notes

- For more information about the Agent, see the *Oracle Essbase Database Administrator's Guide*.
- Make sure you have enough open file descriptors configured in the operating system to accommodate whatever value you set for AGTSVRCONNECTIONS.
- If you set the value of AGTSVRCONNECTIONS to greater than the value set in AGENTTHREADS, Essbase interprets the value as equal to the value of AGENTTHREADS.

Example

```
AGTSVRCONNECTIONS 7
```

Increases the maximum number of simultaneous connections between Agent and server, from the default of 5 to 7.

See Also

[“AGENTTHREADS” on page 388](#)

APPMAXLOGFILESIZE

Sets the maximum size of application log files (*appname.log*).

Syntax

```
APPMAXLOGFILESIZE n
```

Where *n* is the file size in bytes:

- Minimum file size is 1 MB (1048576 bytes). User-specified values less than the minimum are not recognized and are reset to 1 MB.
- Maximum file size is 2 GB (2147483647 bytes). User-specified values greater than the maximum are not recognized and are reset to 2 GB.
- If no value is specified, the default value of 2 GB (2147483647 bytes) is used.

Description

This parameter enables the user to specify the maximum size for application log files.

Application log files are located in *MIDDLEWARE_HOME/user_projects/epmsystem1/diagnostics/logs/essbase/essbase_0/app/appname/appname* or in *ARBORPATH/app/appname*, depending on the value of the *DEFAULTLOGLOCATION* configuration parameter.

The current log file is *appname.log*. When maximum log file size is reached, the file is renamed *appname.log.n* (for example, *appname.log.0*, *appname.log.1*, and so on), and a new *appname.log* file is created.

Example

```
APPMAXLOGFILESIZE 1500000
```

Sets the maximum Agent log file size to 1,500,000 bytes.

APSRESOLVER

Specifies the Oracle Hyperion Provider Services server to use for name resolution, which enables connections to be made using logical Essbase cluster names.

Syntax

```
APSRESOLVER APSur1 [;APSur1]
```

Where *APSur1* is the URL to a Provider Services server, in this format:

```
http[s]://host:port/contextRoot
```

Description

This configuration setting enables the use of logical Essbase cluster names instead of the Essbase URL (for example, *http[s]://host:port/aps/Essbase?ClusterName=logicalName&Secure=yesORno*) during the login process.

When logging in to an Essbase Server, if the server name specified is not a URL, the Essbase client treats the name as a logical name. The Provider Services server specified in APSRESOLVER then resolves the logical name to a physical host.

Notes

- Multiple URLs can be given, delimited by semicolons
- Use `https://` for SSL
- The logical name must be of the form *name:secure*.
- If Provider Services cannot resolve the logical name, the name is treated as a physical name.
- On successful resolution of the logical name of a standalone Essbase Server, the mapping is cached for 5 minutes (not configurable). Subsequent attempts to log in on the same API handle are resolved by the cache. After 5 minutes, the cache entry is discarded and Oracle Hyperion Provider Services is used to resolve the logical name.
- This setting applies only for server-to-server communication (Essbase and C API). For client-to-server communication (Java API), use the `essbase.properties` file. See the *Oracle Hyperion Provider Services Administration Guide* and the *Oracle Hyperion Enterprise Performance Management System High Availability and Disaster Recovery Guide*.

Examples

```
http://qtfsvr1:1234/aps
```

```
http://qtfsvr1:1234/aps;http://qtfsvr2:1234/aps
```

```
https://qtfsvr1:1234/aps
```

```
https://qtfsvr1:1234/aps;http://qtfsvr2:1234/aps
```

ASOLOADBUFFERWAIT

Specifies the maximum amount of time (in seconds) Essbase waits for aggregate storage cache resources to become available in order to process load buffer operations. If cache resources do not become available within the specified amount of time, Essbase aborts the load buffer operation.

This setting applies to the creation of aggregate storage data load buffers with the `wait_for_resources` option, and applies to allocations, custom calculations, and lock and send operations.

This setting applies only to aggregate storage databases.

Syntax

```
ASOLOADBUFFERWAIT [appname [dbname]] n
```

- *appname*—Optional. Specifies the application for which the wait for resources option 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 wait for resources option is to be set.

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

- *n*—Specifies the maximum number of seconds Essbase waits for cache resources to become available.

The default value is 10 seconds.

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

Example

```
ASOLOADBUFFERWAIT ASOSamp Sample 20
```

Sets 20 seconds as the maximum wait time for cache resources to become available on the ASOSamp.Sample database.

See Also

[Alter Database \(Aggregate Storage\) MaxL statement](#)

ASOSAMPLESIZEPERCENT

Specifies the number of cells sampled from the input-level data. The sampled data is used to estimate the size of aggregate views. Larger sample sizes enable Essbase to make increasingly accurate estimates of average view sizes. View selection using a larger sample size enables Essbase to more closely meet the stop size.

Sample sizes are specified as a percentage of input-level data.

Syntax

```
ASOSAMPLESIZEPERCENT [appname [dbname]] n
```

- *appname*—Optional. Application for which sampled data 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 sampled data is to be set.

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

- *n*—A value ranging from 0.0 to 100.0, representing a percentage of input-level cells that are to be used for the aggregate storage cell sample.

To calculate the number of sample cells, multiply the number of input-level cells by the percentage specified in *n*. The default, and minimum, sample size is 1 million (1,000,000) cells.

Note: For databases that have 1 million or more cells, if the percentage specified results in a sample size of fewer than 1 million cells, the setting is ignored and Essbase uses 1 million cells. For databases that have fewer than 1 million cells, the sample size is the same size as the database.

Performance Impact

Estimates using larger sample sizes take longer to complete, which has a potentially significant performance impact on view selection. A recommendation for a database of greater than 100 million input-level cells is to start with a small setting such as 1 (for 1%). Slowly increase this setting until the preferred trade-off between view selection performance and accuracy is reached.

To gauge the accuracy of view size estimates for aggregate views that have been built, use the following MaxL command:

```
query database appname.dbname list existing_views
```

Compare the values in the columns named *size_ratio_estimate* and *size_ratio_actual*. The accuracy of each view size estimate differs for each aggregate view.

Example

```
ASOSAMPLESIZEPERCENT ASOsamp.Sample 1
```

AUTHENTICATIONMODULE

Enables Essbase to use the Oracle's Hyperion® Shared Services security platform for external authentication.

When you run Oracle's Hyperion Enterprise Performance Management System Configurator, Essbase is automatically registered with Shared Services (unless you select the option to deploy Essbase in standalone mode) and this setting is automatically added to *essbase.cfg*.

Syntax

```
AUTHENTICATIONMODULE CSS
```

Notes

- You must restart Essbase Server to initialize the changes.

- Shared Services must be running before you restart Essbase Server, so that Essbase can find the URL to Shared Services.

CALCCACHE

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

This setting does not apply to aggregate storage databases.

Syntax

```
CALCCACHE TRUE | FALSE
```

- TRUE—Essbase uses a calculator cache when calculating the database. The default is TRUE.
- 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 your calculation performance. The size of the performance improvement depends on your 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 SETCACHE command in a calculation script and the CALCCACHE {HIGH | DEFAULT | LOW} settings in the `essbase.cfg` file.

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

- Your 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).

Notes

For detailed information on setting the size of your calculator cache, see the *Oracle Essbase Database Administrator's Guide*.

Example

```
CALCCACHE TRUE  
CALCCACHE FALSE
```

Note: In `essbase.cfg`, the parameter is not followed by a semicolon; in a calculation script, the parameter must be followed by a semicolon.

See Also

[SET CACHE](#) (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 n
```

CALCCACHEHIGH is the level and *n* is the maximum calculator cache size, in bytes, that a user can choose to use during calculation. The maximum calculator cache size that you can specify is 200,000,000 bytes.

Description

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

For detailed information on setting the size of your calculator cache, see the *Oracle Essbase Database Administrator's Guide*.

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

Notes

- In `essbase.cfg`, a setting parameter is not followed by a semicolon; in a calculation script, a parameter must be followed by a semicolon.
- For detailed information on setting the size of your calculator cache, see the *Oracle Essbase Database Administrator's Guide*.

Example

Assume the `essbase.cfg` file contains these settings:

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

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.

See Also

[“CALCCACHEDEFAULT” on page 398](#)

[“CALCCACHELOW” on page 399](#)

[SET CACHE](#) (calculation script command)

CALCCACHEDEFAULT

Sets default value for the calculation script [SET CACHE](#) command.

This setting does not apply to aggregate storage databases.

Syntax

```
CALCCACHEDEFAULT n
```

CALCCACHEDEFAULT is the level and *n* is the size for the level, default in this example, the default calculator cache size, in bytes.

If you do not set the value of DEFAULT, Essbase uses a default value of 200,000 bytes.

Description

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

For detailed information on setting the size of your calculator cache, see the *Oracle Essbase Database Administrator's Guide*.

You can specify whether Essbase uses a calculator cache by default using the CALCCACHE setting in the `essbase.cfg` file. If required during a calculation, override this default setting using the [SET CACHE](#) command in a calculation script.

Notes

- In `essbase.cfg`, a parameter is not followed by a semicolon; in a calculation script, a parameter must be followed by a semicolon.
- For detailed information on setting the size of your calculator cache, see the *Oracle Essbase Database Administrator's Guide*.

Example

Assume the `essbase.cfg` file contains these settings:

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

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.

See Also

[“CALCCACHEHIGH” on page 397](#)

[“CALCCACHELOW” on page 399](#)

[SET CACHE](#) (calculation script command)

CALCCACHELOW

Sets the HIGH, DEFAULT, and LOW values for the calculation script [SET CACHE](#) command.

This setting does not apply to aggregate storage databases.

Syntax

```
CALCCACHELOW n
```

CALCCACHELOW is the level and *n* is the minimum calculator cache size, in bytes, that a user can choose to use during calculation.

Description

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

For detailed information on setting the size of your calculator cache, see the *Oracle Essbase Database Administrator's Guide*.

You can specify whether Essbase uses a calculator cache by default using the CALCCACHE setting in the `essbase.cfg` file. If required during a calculation, override this default setting using the [SET CACHE](#) command in a calculation script.

Notes

- In `essbase.cfg`, a parameter is not followed by a semicolon; in a calculation script, a parameter must be followed by a semicolon.
- For detailed information on setting the size of your calculator cache, see the *Oracle Essbase Database Administrator's Guide*.

Example

Assume the `essbase.cfg` file contains these settings:

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

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.

See Also

[“CALCCACHEHIGH” on page 397](#)

[“CALCCACHEDEFAULT” on page 398](#)

`SET CACHE` (calculation script command)

CALCLIMITFORMULARECURSION

When set to true, prevents the server from going beyond 31 formula execution levels.

Syntax

```
CALCLIMITFORMULARECURSION TRUE | FALSE
```

- **TRUE**—Imposes a limit of 31 on the number of formula execution levels.
- **FALSE**—Imposes no limit on the number of formula execution levels. The default setting is **FALSE**.

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). By default, Essbase does not limit the number of formula execution levels. However, 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 31 execution levels and `CALCLIMITFORMULARECURSION` is set to **TRUE**, Essbase stops processing that formula and writes error messages in the application log. If a formula reaches 31 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

If you added a member named "Payroll Share In Similar Markets" to Sample Basic and used the following formula to calculate it, you would get a recursion error.

```
IF (@ISUDA(Market, "Major Market"))
    Payroll / @SUMRANGE(Payroll, @UDA(Market, "Major Market"));
ELSEIF (@ISUDA(Market, "Small Market"))
    Payroll / @SUMRANGE(Payroll, @UDA(Market, "Small Market"));
ENDIF;
```

CALCLOCKBLOCK

Sets the HIGH, DEFAULT, and LOW values for the calculation script [SET LOCKBLOCK](#) command, which specifies the maximum number of blocks that Essbase can fix (get addressability to) when calculating one block.

This setting does not apply to aggregate storage databases.

Syntax

```
CALCLOCKBLOCKHIGH | CALCLOCKBLOCKDEFAULT | CALCLOCKBLOCKLOW n
```

Where HIGH, DEFAULT, and LOW are levels:

- HIGH—Maximum number of blocks that a user can choose to fix concurrently when one data block is calculated. Maximum: half the number of blocks that fit into the data cache.
- DEFAULT—Default number of blocks that can be fixed concurrently.
- LOW—Minimum number of blocks that a user can choose to fix concurrently.
- *n*—Integer value for each level, representing the total number of blocks that can be locked concurrently.

Description

CALCLOCKBLOCK specifies the number of blocks that can be fixed at each level of the SET LOCKBLOCK HIGH | DEFAULT | LOW calculation script command.

When a block is calculated, Essbase fixes (gets addressability to) the block along with the blocks containing its children. Essbase calculates the block and then releases it along with the blocks containing its children. By default, Essbase allows up to 100 blocks to be fixed concurrently when calculating a block. This is sufficient for most database calculations. However, you may want to set a number higher than 100 if you are consolidating very large numbers of children in a formula calculation. This ensures that Essbase can fix all the required blocks when calculating a data block and that performance will not be impaired.

Notes

- For more information on data blocks, see the *Oracle Essbase Database Administrator's Guide*.
- The maximum you can specify for CALCLOCKBLOCK is half the number of blocks that fit into the data cache. If you specify a number great than this, Essbase defaults to a number equal to half the number of blocks that fit into the data cache.
- You can calculate the number of blocks that fit into the data cache by dividing the data cache size (in bytes) by the block size (in bytes). Values for the data cache size and the block size are available in Administration Services.

Example

If the `essbase.cfg` file contains the following settings:

```
CALCLOCKBLOCKHIGH 500
CALCLOCKBLOCKDEFAULT 200
CALCLOCKBLOCKLOW 50
```

Then you can use the following SET LOCKBLOCK setting commands in a calculation script:

```
SET LOCKBLOCK HIGH;
```

Essbase can fix up to 500 data blocks when calculating one block.

```
SET LOCKBLOCK DEFAULT;
```

Essbase can fix up to 200 data blocks when calculating one block.

```
SET LOCKBLOCK LOW;
```

Essbase can fix up to 50 data blocks when calculating one block.

Note: In `essbase.cfg`, a parameter is not followed by a semicolon; in a calculation script, a parameter must be followed by a semicolon.

See Also

[SET LOCKBLOCK](#) (calculation script command)

CALCMODE

Enables global setting of formula execution mode.

This setting does not apply to aggregate storage databases.

Syntax

```
CALCMODE [application_name [database_name]] [BLOCK | BOTTOMUP]
```

- *application_name*—Optional. If you specify an application, all the databases in that application are affected by the CALCMODE setting. If you leave out the application and database name parameters, the CALCMODE setting applies to the entire server.

- *database_name*—Optional. If you specify an application and database, the database you specify is affected by the CALCMODE setting. If you do not specify an application with the database, the CALCMODE setting will fail.
- BLOCK—Turns on block calculation mode.
- BOTTOMUP—Turns on bottom-up calculation mode.

Description

CALCMODE allows you to set the calculation mode at the server, application, or database level instead of indicating it in a calculation script using @CALCMODE. For more information, see the calculator command entry for @CALCMODE in the *Oracle Essbase Technical Reference*

Example

```
CALCMODE BLOCK
```

Turns on block calculation mode for all databases and applications in the server.

See Also

[@CALCMODE](#) function

CALCNOTICE

Sets the HIGH, DEFAULT, and LOW values for the [SET NOTICE](#) calculation command, which displays completion notices about the progress of the calculation.

This setting does not apply to aggregate storage databases.

Syntax

```
CALCNOTICEHIGH | CALCNOTICEDEFAULT | CALCNOTICELOW n
```

where HIGH, DEFAULT, and LOW are levels.

- HIGH—Maximum number of completion notices that a user can choose to display.
- DEFAULT—Default number of completion notices.
- LOW—Minimum number of completion notices that a user can choose to display.
- *n*—Integer value for each level. It represents the number of notices to be displayed at set intervals during the calculation.

Description

CALCNOTICE defines the values for each of the three levels of the [SET NOTICE](#) calculation command.

SET NOTICE HIGH | DEFAULT | LOW provides completion notices during a calculation. The frequency and number of completion notices depends on the level specified.

The interval between notices is approximate. Essbase measures the interval by taking the number of data blocks already calculated as a percentage of the total number of possible data blocks in your database.

For partial calculations and calculations with multiple passes through your database, the interval between completion notices is approximate.

Notes

- The intervals between completion notices are approximate.
- Completion notices do not significantly reduce the calculation performance, except when used with a very small database.

Example

If you use the following settings in the `essbase.cfg` file:

```
CALCNOTICEHIGH 50  
CALCNOTICEDEFAULT 20  
CALCNOTICELOW 5
```

Then SET NOTICE commands in a script produce the following results:

```
SET NOTICE HIGH;
```

Displays 50 completion notices at 2% intervals.

```
SET NOTICE DEFAULT;
```

Displays 20 completion notices at 5% intervals.

```
SET NOTICE LOW;
```

Displays 5 completion notices at 20% intervals.

Note: In `essbase.cfg`, a parameter is not followed by a semicolon; in a script, a parameter must be followed by a semicolon.

See Also

[SET NOTICE](#) (calculation command)

CALCOPTFRMLBOTTOMUP

Specifies whether Essbase optimizes the calculation of complex formulas on sparse dimensions in large database outlines. If enabled, Essbase performs a bottom-up calculation on formulas that would otherwise require a top-down calculation.

This setting does not apply to aggregate storage databases.

Syntax

```
CALCOPTFRMLBOTTOMUP TRUE | FALSE
```

- TRUE—Optimizes the calculation of formulas on sparse dimensions in large database outlines by forcing a bottom-up calculation.
- FALSE—Does not force a bottom-up calculation for formulas on sparse dimensions in large database outlines. The default is FALSE.

Description

This setting tells Essbase whether to optimize the calculation of formulas on sparse dimensions in large database outlines, so that you can efficiently use [CALC ALL](#) and [CALC DIM](#) commands to calculate the database.

You can override the `CALCOPTFRMLBOTTOMUP` `essbase.cfg` setting by using the `SET FRMLBOTTOMUP` command in a calculation script.

Notes

- For information on complex formulas and top-down calculations, see the *Oracle Essbase Database Administrator's Guide*.
- Forcing a bottom-up calculation on a formula may produce results that are inconsistent with a top-down calculation if:
 - The formula contains complex functions (for example, range functions)
 - The formula's dependencies are not straightforward
- Before using the `CALCOPTFRMLBOTTOMUP` setting in a production environment, be sure to check the validity of calculation results produced when the setting is enabled (set to TRUE).
- The `SET CREATENONMISSINGBLK` calculation command can force top-down calculations, regardless of the value of the `CALCOPTFRMLBOTTOMUP` setting.

Example

```
CALCOPTFRMLBOTTOMUP TRUE
```

See Also

[SET FRMLBOTTOMUP](#) (calculation command)

[SET CREATENONMISSINGBLK](#) (calculation command)

CALCREUSEDYNALCBLOCKS

Controls whether dynamically calculated values are re-used during retrievals.

This setting does not apply to aggregate storage databases.

Syntax

```
CALCREUSEDYNALCBLOCKS TRUE | FALSE
```

- TRUE—Default value. Dynamically calculated values are re-used.

- FALSE—Dynamically calculated values are not re-used.

Description

By default, Essbase re-uses dynamically calculated values during retrievals. This can speed up retrievals that involve a large number of dynamically calculated blocks that are each required to compute several other blocks, such as when there is a large hierarchy of sparse Dynamic Calc members. However, a large dynamic calculator cache size or a large value for the `CALCLOCKBLOCK` may adversely affect the retrieval performance when this method is used. In such cases, `CalcReuseDynCalcBlocks` should be set to `FALSE`.

Example

```
CALCREUSEDYNCALCBLOCKS TRUE  
CALCREUSEDYNCALCBLOCKS FALSE
```

CALCPARALLEL

Enables parallel calculation, defining the number of processing threads.

Syntax

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

- *appname*—Optional. Specifies that parallel calculation applies to all databases on the named 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*—Optional. Specifies that parallel calculation applies only to the database named. 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 databases on the Essbase Server.
- *n*—A required parameter that specifies the number of threads to be made available for parallel calculation.
 - For block storage on 32-bit platforms, an integer from 1-4. For block storage on 64-bit platforms, an integer between 1-8. The default value, 1, specifies serial calculation: no parallel calculation takes place.
 - For aggregate storage, an integer from 1-8, 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 Server to initialize any change to the configuration file.

Description

This setting enables 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

- For detailed information about how Essbase performs parallel calculation with block storage databases, see the *Oracle Essbase Database Administrator's Guide*.
- With block storage databases, if your outline generates many empty tasks, thus reducing opportunities for parallel calculation, consider setting the `CALCTASKDIMS` configuration setting to increase the number of tasks and to decrease the size of each task identified for parallel calculation. See the *Oracle Essbase Database Administrator's Guide* 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. For details, see the *Oracle Essbase Database Administrator's Guide* for information about aggregate storage cache.

Example

```
CALCPARALLEL 3
```

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

See Also

[“CALCTASKDIMS” on page 407](#)

[SET CALCPARALLEL](#) calculation command

[SET CALCTASKDIMS](#) calculation command

CALCTASKDIMS

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

This setting does not apply to aggregate storage databases.

Syntax

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

- *appname*—Optional. `CALCTASKDIMS` applies to all databases on the named 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*—Optional. Database 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 databases on the Essbase Server.

- *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.

The default value, 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.

Note: Values less than 0 treated differently than [SET CALCTASKDIMS](#) configuration setting.

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

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

- A number of features are affected by parallel calculation. See the *Oracle Essbase Database Administrator's Guide* for a list of these effects and for detailed information about how Essbase performs parallel calculation.
- Use this configuration setting only if your outline generates many empty tasks, thus reducing opportunities for parallel calculation. See the *Oracle Essbase Database Administrator's Guide* 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” on page 406](#)

[SET CALCPARALLEL](#) calculation command

[SET CALCTASKDIMS](#) calculation command

CCTRACK

Controls whether exchange rates are tracked as Essbase calculates currency conversions.

This setting does not apply to aggregate storage databases.

Syntax

CCTRACK TRUE | FALSE

- TRUE—Exchange rates are tracked while conversions are calculated. The default value is TRUE.
- FALSE—Turns off the tracking system.

Description

CCTRACK controls whether exchange rates are tracked while Essbase calculates currency conversions. Tracking exchange rates has the following advantages:

- Allows conversion to occur at report time through the Spreadsheet Add-in or the Report Writer
- Allows you to convert a converted currency back to its original, local rate using the [CCONV](#) command
- Prevents data inaccuracies due to accidental reconversion of data during a calculation.

After loading data, you can clear the tracked exchange rates for the new data using the [CLEARCCTRACK](#) command. During a calculation, you can enable or disable CCTRACK using the [SET CCTRACKCALC](#) calculation command.

Notes

- When CCTRACK is turned on the following restrictions apply:
 - If you are using currency partitions, you cannot use a CCONV command with a FIX statement to convert a subset of a currency partition (a calculation script attempting such a FIX will not validate).
 - If you are not using currency partitions, you must use CCONV with a FIX statement.
- Setting CCTRACK to FALSE turns off the tracking system with the following results:
- The CCONV assumes that the data is unconverted (in local currency). If you accidentally run the CCONV command multiple times on the same data, the resulting data will be inaccurate.
- Similarly, the currency report options assume that the data is unconverted (in local currency). If the data has already been converted in the database, it is reconverted at report time, resulting in inaccurate data.
- The restrictions on using the [FIX...ENDFIX](#) and [DATACOPY](#) commands in currency conversions do not apply. For example, if you are using currency partitions, you can now use the FIX command with the CCONV command to calculate a subset of a currency partition. If you are not using currency partitions, you can use CCONV without a FIX statement.

Example

```
CCTRACK TRUE
```

See Also

[CCONV](#) (calculation command)

[SET UPTOLOCAL](#)

[SET CCTRACKCALC](#) (calculation command)

[CLEARCCTRACK](#) (calculation command)

CLEARLOGFILE

Determines whether the Essbase Server and application logs are overwritten.

Syntax

```
CLEARLOGFILE TRUE | FALSE
```

- TRUE—Overwrites the Essbase Server and application logs.
- FALSE—Appends to the existing logs. The default setting is FALSE.

Description

CLEARLOGFILE determines whether the Essbase Server log (*essbase.log*) is overwritten whenever Essbase Server is restarted and whether the application log (*application_name.log*) is overwritten whenever the application is restarted.

Notes

This setting affects both the application and Essbase Server logs. Essbase logs the error to the appropriate files automatically.

Examples

Example 1

If Essbase logs an application message and this setting is in effect:

```
CLEARLOGFILE TRUE
```

Essbase logs the message in the *application_name.log* file in the application directory: *ARBORPATH\app\application_name*, where *application_name* is the name of the current application. The contents of this log are replaced with new entries each time the application is started.

Example 2

If Essbase logs a server message and this setting is in effect:

```
CLEARLOGFILE FALSE
```

Essbase logs the message in the `essbase.log` file in the directory pointed to by `ARBORPATH`, appending the existing file.

See Also

[“SSLOGUNKNOWN” on page 503](#)

CLIENTPREFERREDMODE

Enables SSL connectivity to Essbase.

Syntax

```
CLIENTPREFERREDMODE SECURE | CLEAR
```

- **SECURE**—Essbase communicates with clients using only SSL.
- **CLEAR**—Client sessions are based on the transport specified in the login API. If the secure transport is specified, then the session uses SSL; otherwise, the session uses clear. The default value is CLEAR.

Description

This setting determines whether Essbase allows only SSL connectivity. It applies only to clients.

Example

```
CLIENTPREFERREDMODE SECURE
```

See Also

[“AGENTSECUREPORT” on page 388](#)

[“ENABLECLEARMODE” on page 439](#)

[“ENABLESECUREMODE” on page 440](#)

[“NETSSLHANDSHAKETIMEOUT” on page 474](#)

[“SSLCIPHERSUITES” on page 502](#)

[“WALLETPATH” on page 515](#)

For information on implementing SSL, see the *Oracle Hyperion Enterprise Performance Management System Security Administration Guide*.

CRASHDUMP

Sets whether Essbase saves a core dump to a file when an abnormal termination of an agent or server process occurs. UNIX only.

Syntax

```
CRASHDUMP TRUE | FALSE
```

- TRUE—Creates a directory containing a core file for each abnormal termination.
- FALSE—No core file is created. This is the default value.

Description

CRASHDUMP helps diagnose abnormal program terminations. For each agent crash, when CRASHDUMP is set to TRUE, Essbase creates a file named *core*. It places the core file in an *ESSBASE.abc* directory under *ESSBASEPATH*, where *abc* displays the date and time. For example:

```
ESSBASE.Mon_Jun_3_18_16_17_2003/core
```

In each instance of a server crash, when CRASHDUMP is set to TRUE, Essbase creates the core file in a directory under *ARBORPATH/app/appName*, where *appName* is the name of the application. The name of the new directory is *ESSSVR.abc*, where *abc* displays the date and time. For example:

```
/EssbaseServer/app/Sample/ESSSVR.Mon_Jun_3_18_16_17_2003/core
```

If the an agent or server process is automatically shut down, the core file contains a core dump of that moment. If an agent or server process is shut down manually, the core file may be empty.

Look for the core file any time you experience abnormal Essbase program terminations. If the file is not empty, provide it to Support and then remove it and its directory from the computer. If the core file is empty, remove it and its directory from the computer.

In normal operations without abnormal terminations, core files are not created.

Example

```
CRASHDUMP TRUE
```

DATA CACHESIZE

Defines the initial value for the data cache size for any new databases that are created after Essbase is restarted. 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

```
DATA CACHESIZE n
```

Where *n* is an integer 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

DATACACHE SIZE specifies, in bytes, kilobytes, megabytes, or gigabytes, the size of the data cache for new databases on the server. The specified value takes effect for all new databases that are created after the server is started. To set or change the data cache size for an individual database, use Administration Services or MaxL. For more information, see the online help or HTML documentation for those components.

Example

```
DATACACHE SIZE 90M
```

Sets the data cache size of all newly created or migrated databases as 90 megabytes.

See Also

[“DATAFILECACHE SIZE” on page 415](#)

[“MEMSCALINGFACTOR” on page 471](#)

DATAERRORLIMIT

Determines the number of records that can be written to an error log during a data load operation.

Syntax

```
DATAERRORLIMIT n
```

Where *n* is the number of records, per data load or dimension build, that can be written to the error log, `dataload.err`. Default: 1000. Maximum: 65,000.

Description

DATAERRORLIMIT determines the number of records that can be written to the error log during data load or dimension build operations.

After the specified number of errors have been recorded, Essbase fails the operation and issues an error message.

Notes

- Essbase logs data load errors in `EAS_HOME\client\dataload.err`.
- Essbase logs dimension build errors in `EAS_HOME\client\dimbuild.err`.
- Messages are still written to the application log unless you set `NOMSGLOGGINGONDATAERRORLIMIT`.

Example

```
DATAERRORLIMIT 1000
```

See Also

[“NOMSGLOGGINGONDATAERRORLIMIT” on page 476](#)

DATAEXPORTENABLEBATCHINSERT

Specifies whether to use the batch-insert method, instead of the default row-insert method, when the [DATAEXPORT](#) calculation command is used to export Essbase data for direct insertion into a relational database.

The DATAEXPORTENABLEBATCHINSERT and DEXPSQLROWSIZE configuration settings apply to block storage databases only.

```
DATAEXPORTENABLEBATCHINSERT TRUE | FALSE
```

- TRUE—Enables batch insert of exported data into a relational database
- FALSE—(Default) Inserts exported data row-by-row into a relational database

Description

When DATAEXPORTENABLEBATCHINSERT is set to TRUE, Essbase determines whether the relational database and the ODBC driver permit batch insert. If they do, Essbase uses the batch-insert method, and, thus, performance is optimized.

Essbase determines the batch size; however, you can control the number of rows (from 2 to 1000) that are inserted at one time by using the DEXPSQLROWSIZE configuration setting.

If Essbase cannot determine whether the relational database and the ODBC driver support batch insert, it uses the row-insert method, and DEXPSQLROWSIZE (if set) is ignored.

When DATAEXPORTENABLEBATCHINSERT is set to FALSE, an INSERT command is called for each row of exported data, and, thus, performance is slowed.

Notes

- If DATAEXPORTENABLEBATCHINSERT is set to TRUE and DEXPSQLROWSIZE is set to 1, batch insert is disabled (as a DEXPSQLROWSIZE setting of 1 inserts one row at a time).
- When using DATAEXPORT to export data for direct insertion into a relational database:
 - The table to which the data is to be written must exist prior to the data export
 - Table and column names cannot contain spaces

See Also

[DATAEXPORT](#) calculation command

[“DEXPSQLROWSIZE” on page 418](#) configuration setting

DATAFILECACHESIZE

Defines the initial value for the data file cache size for all new databases that are created or migrated. The data file cache is a buffer in memory that holds data files. Essbase allocates this memory during data load, calculation, and retrieval operations, as needed.

This setting does not apply to aggregate storage databases.

Syntax

```
DATAFILECACHESIZE n
```

Where *n* is an integer expressed in bytes (B), kilobytes (K), megabytes (M), or gigabytes (G)

- Minimum value: 8 megabytes (8 M)
- Maximum value: 2 gigabytes (2 G)
- Default value: 32 megabytes (32 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

DATAFILECACHESIZE specifies, in bytes, kilobytes, megabytes, or gigabytes, the size of the data file cache for new databases on the server. The specified value takes effect for all new databases that are created after the server is started. To set or change the data file cache size for an individual database, use Administration Services or MaxL. For more information, see the online help or HTML documentation for those components.

Notes

If this setting is added to the `essbase.cfg` file while Essbase is running, the effect begins after a restart.

Example

```
DATAFILECACHESIZE 800M
```

Defines the data file cache size of all subsequently created databases as 800 megabytes.

See Also

[“DATA CACHESIZE” on page 412](#)

[“MEMSCALINGFACTOR” on page 471](#)

DEFAULTLOGLOCATION

Sets the location of application log files

Syntax

DEFAULTLOGLOCATION TRUE | FALSE

- TRUE—(the default value). The logs are written to one of three locations, based upon the following:
 - If the `HYPERION_LOGHOME` environment variable is set, then the log files are written to the `HYPERION_LOGHOME` directory.
 - If the `HYPERION_LOGHOME` environment variable is *not* set, then the log files are written to `EPM_ORACLE_INSTANCE/diagnostics/logs/essbase`
 - If the `EPM_ORACLE_INSTANCE` environment variable is *not* set, then the log files are written to `HYPERION_HOME/logs/essbase`
- FALSE—The logs are written to one of two locations, based upon the following:
 - Log files for the agent (ESSBASE) are written to `$ARBORPATH/<logfile>`
 - Log files for the server (ESSSVR) are written to `$ARBORPATH/app/<appname>/<logfile>`

Description

The `DEFAULTLOGLOCATION` setting sets the location of application files. TRUE is the default value.

Example

```
DEFAULTLOGLOCATION FALSE
```

DELAYEDRECOVERY

Determines whether Essbase delays free space recovery after an application crashes or terminates abnormally.

This setting does not apply to aggregate storage databases.

Syntax

DELAYEDRECOVERY [*appname*] TRUE | FALSE

- *appname*—Optional. The name of an application to which this setting should apply. If omitted, all applications are affected.
- TRUE—Essbase delays freespace recovery
- FALSE—Essbase does not delay freespace recovery.

Description

This setting controls whether Essbase delays freespace recovery.

Database recovery takes place any time you load an application that has just crashed or terminated abnormally. Essbase does not perform free space recovery automatically because it

is the most expensive part of database recovery. You must either trigger freespace recovery explicitly or change the default setting so that Essbase will recover free space automatically.

Example

```
DELAYEDRECOVERY TRUE
```

Essbase delays freespace recovery.

See Also

[Alter Database](#)<DBS-NAME> recover freespace, which is the statement you use to explicitly recover freespace.

DELIMITEDMSG

Separate fields when writing log files, using the default (~) character.

Syntax

```
DELIMITEDMSG [TRUE | FALSE]
```

Description

DELIMITEDMSG specifies whether Essbase Server and application logs are delimited in Essbase. If set to TRUE, and no value for “[DELIMITER](#)” on page 417 is supplied, the default tilde (~) is used to delimit fields. If set to FALSE, any value specified in DELIMITER is ignored, and no special delimiter is used for logs.

Example

```
DELIMITEDMSG TRUE  
DELIMITER *
```

Essbase produces logs that use the asterisk (*) symbol as a delimiter between fields in a log.

See Also

[“DELIMITER” on page 417](#)

DELIMITER

Delimits Essbase Server and application logs using one of five allowed symbols.

Syntax

```
DELIMITER [~ | ^ | * | : | & ]
```

Description

DELIMITER specifies which of five symbols that Essbase will use to delimit fields in logs. DELIMITER is ignored unless DELIMITEDMSG TRUE is also present in the configuration file.

Example

```
DELIMITEDMSG TRUE
DELIMITER *
```

Essbase produces logs that use the asterisk (*) symbol as a delimiter between fields in a log.

See Also

[“DELIMITEDMSG” on page 417](#)

DEXPSQLROWSIZE

When the DATAEXPORT calculation command is used to export data directly into a relational database and when the batch-insert method is used, the DEXPSQLROWSIZE configuration setting allows you to specify the number of rows to be inserted at one time.

To enable batch insert, set the DATAEXPORTENABLEBATCHINSERT configuration setting to TRUE. Essbase determines whether the relational database and the ODBC driver permit batch insert. If they do, Essbase determines the batch size unless you set DEXPSQLROWSIZE. If Essbase cannot determine whether the relational database and the ODBC driver support batch insert, it uses the row-insert method, and DEXPSQLROWSIZE (if set) is ignored.

The DEXPSQLROWSIZE and DATAEXPORTENABLEBATCHINSERT configuration settings apply to block storage databases only.

```
DEXPSQLROWSIZE [appname [dbname]] n
```

- *appname*—Optional. Specifies the application for which to set the number of rows to be inserted at one time.

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 to set the number of rows to be inserted at one time.

If you specify a value for *dbname* but do not specify a value for *appname*, your specification is ignored, and data associated with logged transactions is archived for all applications and databases on Essbase Server.

- *n*—The number of rows in the batch (from 2 to 1000).

Notes

- If DATAEXPORTENABLEBATCHINSERT is set to TRUE and DEXPSQLROWSIZE is set to 1, batch insert is disabled (as a DEXPSQLROWSIZE setting of 1 inserts one row at a time).
- When using DATAEXPORT to export data for direct insertion into a relational database:
 - The table to which the data is to be written must exist prior to the data export

- Table and column names cannot contain spaces

Example

```
DEXPSQLROWSIZE Sample Basic 300
```

Specifies a 300-record batch size for data exported from Sample.Basic to a relational database using DATAEXPORT.

```
DEXPSQLROWSIZE Sample 500
```

Specifies a 500-record batch size for data exported from any database within the Sample application to a relational database using DATAEXPORT.

See Also

[DATAEXPORT](#)

[“DATAEXPORTENABLEBATCHINSERT” on page 414](#)

DIMBUILDERRORLIMIT

Determines the number of records that can be written to an error log during a dimension build operation.

Syntax

```
DIMBUILDERRORLIMIT n
```

Where *n* is the number of records, per dimension build, that can be written to the error log, `dimbuild.err`. Default: 20,000. Maximum: 65,000.

Description

DIMBUILDERRORLIMIT determines the number of records that can be written to the error log during dimension build operations.

After the specified number of errors have been recorded, Essbase no longer records any more errors, but continues the dimension build process.

Notes

- Essbase logs dimension build errors in `EAS_HOME\client\dimbuild.err`.
- Essbase logs data load errors in `EAS_HOME\client\dataload.err`.

Example

```
DIMBUILDERRORLIMIT 40000
```

See Also

[“DATAERRORLIMIT” on page 413](#)

DIMBUILDSTATSINTERVAL

When performing a cube deployment operation in Oracle Essbase Studio, DIMBUILDSTATSINTERVAL specifies the number of records to process before reporting on dimension build progress. Progress information is displayed in the Essbase application console.

Load status information is written to the Essbase log file.

The default value is 20000, meaning that dimension build progress information is updated in the build status window after each 20000 records is processed.

Syntax

```
DIMBUILDSTATSINTERVAL [n]
```

n—Required. An integer specifying the number of records to process before updating the dimension build progress information in the Essbase application console.

Example

```
DIMBUILDSTATSINTERVAL 20000
```

If there are 50000 records to process in the data source, and DIMBUILDSTATSINTERVAL is defined at 20000, Essbase shows the dimension build progress in the Essbase application console after processing 20000 records, and then 40000 records.

DIRECTIO

Sets the file access mode to direct I/O instead of the default buffered I/O. Applies only to new databases or to databases migrated from Release 6.2 or earlier.

This setting does not apply to aggregate storage databases.

Syntax

```
DIRECTIO TRUE | FALSE
```

- TRUE—Direct I/O is used, when possible, for newly created or migrated databases.
- FALSE—This is the default. Buffered I/O is used for newly created or migrated databases.

Description

For each database, a security file setting tells Essbase whether to use buffered or direct I/O when it accesses the database. By default, when Essbase creates a new database or migrates one from release 6.2 or earlier, it sets this I/O access mode setting to buffered I/O. You can specify the DIRECTIO TRUE configuration setting to change the default setting for new or migrated databases to be direct I/O.

To alter the I/O access mode setting for a database thereafter, use Administration Services or MaxL.

Notes

- Effective use of direct I/O requires a larger index cache than is needed for buffered I/O. See the *Oracle Essbase Database Administrator's Guide* section on sizing caches for details.
- On operating systems and file systems that do not support direct I/O, buffered I/O is used regardless of the setting in the security file.

Example

```
DIRECTIO TRUE
```

When Essbase is restarted, the file access mode is set to direct I/O for new databases and databases migrated from release 6.2 or earlier.

DISABLEREPLMISSINGDATA

Instructs Essbase not to replicate #MISSING values to the target partition, thus improving performance, potentially with less accurate data.

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

Syntax

```
DISABLEREPLMISSINGDATA [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—#MISSING values are not replicated to the target for those applications and databases specified through the *apname* and *dbname* parameters.
- FALSE—(Default value) #MISSING values are replicated to the target for those applications and databases specified through the *apname* and *dbname* parameters.

Notes

This setting applies only to replicated partitions on block storage databases.

When #MISSING data is not replicated a warning message is logged in the application log file.

Example

Assume a partition exists from Sample1.Basic (source) to Sample2.Basic (target). To prevent replication of #MISSING data, add the following settings to `essbase.cfg`.

```
DISABLEREPLMISSINGDATA Sample1 Basic TRUE
DISABLEREPLMISSINGDATA Sample2 Basic TRUE
```

DISKVOLUMES

Defines the volumes that can be used to store multiple index and data files, and the amount of space that those volumes can occupy.

For new files, disk volume settings become effective after the database is restarted. Previously existing files and volumes are not affected.

This setting does not apply to aggregate storage databases.

Syntax

```
DISKVOLUMES [volume_name] [disk_space]...
```

- *volume_name*—The name of the directory where a hard disk is mounted.
 - On Windows, *volume_name* is a letter corresponding to a disk drive.
 - On UNIX, *volume_name* is a UNIX file path that you must specify up to the directory that you are using for Essbase. Do not specify the `/app` directory; Essbase appends `/app` automatically.

Note: Use only valid volume types. Do not use NFS, floppy, CD-ROM, or network drives.

If you do not specify any disk volumes, Essbase uses only the volume where the `ARBORPATH` directory resides.

- *disk_space*—The maximum number of bytes allocated to the volume.
 - Specify this setting in bytes, kilobytes (K), megabytes (M) or gigabytes (G). Do not use commas or spaces. Avoid decimals (such as 2.5G).
 - The value is read as bytes.
 - The maximum value is 2147483648 (2^{31}).
 - If you need to specify a value over 2^{31} , you must use a qualifier (K, M, or G); for example, 2000G.

If you enter a value *with* a qualifier (K, M, or G), the acceptable value range per volume is 0 to 2 terabytes. Do not exceed this amount by specifying, for example, 50000G.

If you specify *volume_name* without specifying *disk_space*, all the disk space on that volume is used, as needed.

If you do not specify *volume_name*, Essbase uses the volume where the `ARBORPATH` directory resides.

DISKVOLUMES, with its values, can be up to 2 kilobytes long. You can specify 64 items per line; for example, `DISKVOLUMES D 5M E 2M C 5G` contains 7 items.

Notes

- Use DISKVOLUMES only if you need backward compatibility with earlier releases, or if you are setting up a large number of databases at the same time with the same DISKVOLUMES value. Otherwise, to set or change disk volumes, use Administration Services or MaxL.
- You can specify disk volume names in any order.
- If you wish to use a volume in the *ARBORPATH* directory, you must specify *ARBORPATH* as one of your parameters. Otherwise, you do not need to specify *ARBORPATH*.

Example

On Windows, the following command causes index and data files to be stored as follows:

```
DISKVOLUMES D 5M E 2M C 5G
```

- The first 5 megabytes on drive D
- The next 2 megabytes on drive E
- The next 5 gigabytes on drive C

On UNIX platforms the following command causes index and data files to be stored as follows::

```
DISKVOLUMES /vol2/essbase 5M /vol3/essbase 2M /vol1/essbase 5G
```

- The first 5 megabytes on volume vol2
- The next 2 megabytes on volume vol3
- The next 5 gigabytes on volume vol1

See Also

[Alter Database \(disk volumes\)](#)

DISPLAYMESSAGELEVEL

Sets the level of messages displayed in the application console. To set the level of messages written to the application *log*, use LOGMESSAGELEVEL.

Syntax

```
DISPLAYMESSAGELEVEL INFO | WARNING | ERROR
```

Where INFO, WARNING, and ERROR are priority levels:

- INFO—All three types of messages are written to the application console. This is the default.
- WARNING—Only Warning and Error messages are written to the application console.
- ERROR—Only error messages are written to the application console. No Warning or Info messages are written to the console.

Notes

This setting affects only the messages displayed in the application console. To control the messages written to the application log, use “[LOGMESSAGELEVEL](#)” on page 465. To set the same level for both the console and the log, use both settings.

Example

```
DISPLAYMESSAGELEVEL WARNING
```

Sets the console message level to Warning. Only Warning and Error messages are displayed in the application console.

See Also

[SETMSGLEVEL](#)

“[LOGMESSAGELEVEL](#)” on page 465

DLSINGLETHREADPERSTAGE

Instructs Essbase to load data using a single thread per processing stage, or to use the thread values specified in the “[DLTHREADSPREPARE](#)” on page 426 and “[DLTHREADSWRITE](#)” on page 427 configuration settings. By working with these three configuration settings you may be able to test and improve data load performance.

You can specify DLSINGLETHREADPERSTAGE for individual databases, 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 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—Tells Essbase **not** to use the values in the “[DLTHREADSPREPARE](#)” on page 426 and “[DLTHREADSWRITE](#)” on page 427 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](#)” on page 426 and “[DLTHREADSWRITE](#)” on page 427 as the numbers of threads to use in the preparation and write stages of data load processing. The default value is FALSE.

Description

The `DLTHREADSPREPARE`, `DLTHREADSWRITE` on page 427, and `DLSINGLETHREADPERSTAGE` on page 424 settings 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. For more information about parallel thread processing in data loads, see the "Optimizing Data Loads" chapter in the *Oracle Essbase Database Administrator's Guide*.

Notes

- While testing thread values for the `DLTHREADSPREPARE` on page 426 and `DLTHREADSWRITE` on page 427 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 the "Optimizing Data Loads" chapter in the *Oracle Essbase Database Administrator's Guide*.

Examples

Example 1

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

Essbase ignores any values specified by `DLTHREADSPREPARE` on page 426 and `DLTHREADSWRITE` on page 427 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` on page 426 and `DLTHREADSWRITE` on page 427 configuration settings for all data bases 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” on page 426 and “DLTHREADSWRITE” on page 427 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” on page 427 settings are included with different thread values for each specific database. Because no “DLTHREADSPREPARE” on page 426 setting is specified, the preparation stage is single-threaded.

See Also

[“DLTHREADSPREPARE” on page 426](#)

[“DLTHREADSWRITE” on page 427](#)

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 DLTHREADSPREPARE 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 DLTHREADSPREPARE, the configuration setting “DLSINGLETHREADPERSTAGE” on page 424 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 produce for preparing the data to be loaded. Specify an integer between 1 and 16 (on 32-bit platforms), or between 1 and 32 (on 64-bit platforms). The default value is 1.

If *n* is greater than the maximum or a negative number, the value is assumed to be 16 (on 32-bit platforms) or 32 (on 64-bit platforms).

Description

The `DLTHREADSPREPARE`, [“DLTHREADSWRITE” on page 427](#), and [“DLSINGLETHREADPERSTAGE” on page 424](#) settings 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. For more information about parallel thread processing in data loads, see the "Optimizing Data Loads" chapter in the *Oracle Essbase Database Administrator's Guide*.

Notes

- You can use another configuration setting, [“DLTHREADSWRITE” on page 427](#), 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 significantly consume 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” on page 424](#) 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” on page 427](#)

[“DLSINGLETHREADPERSTAGE” on page 424](#)

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 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*.
- *n*—The number of threads the data load process may produce for writing data blocks to the disk. Specify an integer between 1 and 16 (on 32-bit platforms), or between 1 and 32 (on 64-bit platforms). The default value is 1.
 - If $n > 16$ (on 32-bit platforms), or a negative number, the value is assumed to be 16.
 - If $n > 32$ (on 64-bit platforms), or a negative number, the value is assumed to be 32.

See Notes below.

Description

The DLTHREADSPREPARE, [“DLTHREADSWRITE” on page 427](#), and [“DLSINGLETHREADPERSTAGE” on page 424](#) settings 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.

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” on page 424](#) must be set to FALSE.

For more information about parallel thread processing in data loads, see the "Optimizing Data Loads" chapter in the *Oracle Essbase Database Administrator's Guide*.

Notes

- You can use another configuration setting, [“DLTHREADSPREPARE” on page 426](#), 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 significantly consume 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](#)” on page 424 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](#)” on page 426

“[DLSINGLETHREADPERSTAGE](#)” on page 424

DYNCALCCACHEBLKRELEASE

Enables Essbase to create a temporary buffer for dynamic calculations in cases where the wait for space in the dynamic calculator cache has exceeded the specified wait time.

This setting does not apply to aggregate storage databases.

Syntax

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

- *appname*—If you supply an application name, the setting applies to all databases within the application. If you do not supply an application name, the setting applies to all applications and databases on the server.
- *dbname*—If you supply a database name, the setting applies only to the database. If you do not also provide an application name, the setting applies to all applications and databases on the server.
- `TRUE`—Tells Essbase to make room available in the dynamic calculator cache by temporarily storing inactive blocks in a separate, compressed-block buffer.
- `FALSE`—This is the default value. Tells Essbase not to find room in the dynamic calculator cache for a different set of blocks. Instead, if allowed by the “[DYNCALCCACHEONLY](#)” on page 435 setting, Essbase attempts to perform calculations on these blocks in memory outside the dynamic calculator cache

Description

Use this setting to tell Essbase to make room available in the dynamic calculator cache, if needed, by compressing inactive blocks from that cache and attempting to temporarily store them in a separate, compressed-block buffer.

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 the dynamic calculator cache may improve retrieval performance by reducing the number of calls to the operating system to do memory allocations. The size of the improvement depends on your database configuration.

Notes

The following sequence of events must occur and settings must be defined before Essbase releases space in the dynamic calculator cache:

- The area allocated in the dynamic calculator cache has reached the maximum allowed (specified by “[DYNCALCCACHEMAXSIZE](#)” on page 433).
- “[DYNCALCCACHEWAITFORBLK](#)” on page 436 is set as TRUE and the wait period specified by “[DYNCALCCACHEBLKTIMEOUT](#)” on page 430 has been reached.
- `DYNCALCCACHEBLKRELEASE` is set to TRUE. Essbase releases an area in the dynamic calculator cache by compressing blocks from this cache and attempting to store them temporarily in a compressed-block buffer. The size of this buffer is defined by the “[DYNCALCCACHECOMPRBLKBUFSIZE](#)” on page 432 configuration setting.

Example

```
DYNCALCCACHEBLKRELEASE TRUE
```

Essbase makes needed space available in the dynamic calculator cache by compressing inactive blocks and temporarily storing them in a dynamic calculator cache compressed-block buffer.

See Also

[“DYNCALCCACHEMAXSIZE” on page 433](#)

[“DYNCALCCACHEWAITFORBLK” on page 436](#)

[“DYNCALCCACHEBLKTIMEOUT” on page 430](#)

[“DYNCALCCACHEONLY” on page 435](#)

[“DYNCALCCACHECOMPRBLKBUFSIZE” on page 432](#)

DYNCALCCACHEBLKTIMEOUT

Specifies maximum time to wait for free space in the dynamic calculator cache.

This setting does not apply to aggregate storage databases.

Syntax

```
DYNCALCCACHEBLKTIMEOUT [appname [dbname]] n
```

- *appname*—If you supply an application name, the setting applies to all databases within the application. If you do not supply an application name, the setting applies to all applications and databases on the server.
- *dbname*—If you supply a database name, the setting applies only to the database. If you do not also provide an application name, the setting applies to all applications and databases on the server.
- *n*—A number of seconds. May or may not include a decimal point. Any number less than 0.001 will be treated as 0.001. The default value is 10 seconds.

Description

Use this setting to specify the maximum number of seconds that Essbase should wait for space in the dynamic calculator cache in order to perform the requested calculation there. If Essbase waits the entire number of seconds specified in this setting, it then checks the [“DYNALCCACHEBLKRELEASE” on page 429](#) setting to determine what to do next:

- To make room in the dynamic calculator cache by temporarily swapping out blocks in the dynamic calculator cache that are inactive
- If [“DYNALCCACHEONLY” on page 435](#) is set to FALSE, to write and calculate the blocks in memory outside the dynamic calculator cache

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 the dynamic calculator cache may improve retrieval performance by reducing the number of calls to the operating system to do memory allocations. The size of the improvement depends on your database configuration.

Notes

- Use the [“DYNALCCACHEBLKRELEASE” on page 429](#) setting to tell Essbase where to store and calculate data blocks containing Dynamic Calc members if the wait for space in the dynamic calculator cache has exceeded the specified wait time.
- The DYNALCCACHEBLKTIMEOUT configuration setting is meaningful only when the [“DYNALCCACHEWAITFORBLK” on page 436](#) configuration setting is set to TRUE.

Example

```
DYNALCCACHEBLKTIMEOUT 20
```

Essbase waits up to 20 seconds for space in the dynamic calculator cache before checking the [“DYNALCCACHEBLKRELEASE” on page 429](#) setting to determine the next step to take before performing the requested calculation.

See Also

[“DYNALCCACHEMAXSIZE” on page 433](#)

[“DYNALCCACHEONLY” on page 435](#)

[“DYNALCCACHEWAITFORBLK” on page 436](#)

[“DYNALCCACHEBLKRELEASE” on page 429](#)

[“DYNALCCACHECOMPRBLKBUFSIZE” on page 432](#)

DYNALCCACHECOMPRBLKBUFSIZE

Specifies the size of a temporary buffer for storing compressed blocks in order to make more space in the dynamic calculator cache.

This setting does not apply to aggregate storage databases.

Syntax

```
DYNALCCACHECOMPRBLKBUFSIZE [appname [dbname]] n
```

- *appname*—If you supply an application name, the setting applies to all databases within the application. If you do not supply an application name, the setting applies to all applications and databases on the server.
- *dbname*—If you supply a database name, the setting applies only to the database. If you do not also provide 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 the compressed block buffer.
 - Default value: 1 megabyte (1M, which is 1,048,576 bytes)
 - 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 (1M)

Description

In order to make space available in the dynamic calculator cache, Essbase uses the value specified by the DYNALCCACHECOMPRBLKBUFSIZE configuration setting to size the dynamic calculator cache compressed-block buffer. Essbase temporarily stores compressed blocks from the dynamic calculator cache into this buffer under the following circumstances:

- The area allocated in the dynamic calculator cache has reached the maximum allowed (specified by [“DYNALCCACHEMAXSIZE” on page 433](#)) and Essbase requires additional space for blocks to be calculated in the current query.
- [“DYNALCCACHEWAITFORBLK” on page 436](#) is set to TRUE and the wait period specified by [“DYNALCCACHEBLKTIMEOUT” on page 430](#) has been reached.
- [“DYNALCCACHEBLKRELEASE” on page 429](#) is set to TRUE, indicating Essbase should release dynamic calculator cache area.

The dynamic calculator cache compressed-block buffer is an area in memory where Essbase compresses and temporarily stores blocks from the dynamic calculator cache to free space for other blocks for other calculations. When space is again available, Essbase decompresses blocks stored in the compressed-block buffer and returns them to the dynamic calculator cache.

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 the dynamic calculator cache may improve retrieval performance by reducing the number of calls to the operating system to do memory allocations. The size of the improvement depends on your database configuration.

Notes

Essbase uses the temporary compressed-block buffer only when the [“DYNALCCACHEBLKRELEASE” on page 429](#) configuration parameter is set to TRUE and the DYNALCCACHECOMPRBLKBUFSIZE setting is greater than 0.

Example

```
DYNALCCACHECOMPRBLKBUFSIZE 1000000
```

Sets 1,000,000 (one million) bytes as the size for the dynamic calculator cache compressed-block buffer.

See Also

[“DYNALCCACHEMAXSIZE” on page 433](#)

[“DYNALCCACHEONLY” on page 435](#)

[“DYNALCCACHEWAITFORBLK” on page 436](#)

[“DYNALCCACHEBLKTIMEOUT” on page 430](#)

[“DYNALCCACHEBLKRELEASE” on page 429](#)

DYNALCCACHEMAXSIZE

Specifies the maximum amount of memory allocated for the dynamic calculator cache.

This setting does not apply to aggregate storage databases.

Syntax

```
DYNALCCACHEMAXSIZE [appname [dbname]] n
```

- *appname*—If you supply an application name, the setting applies to all databases within the application. If you do not supply an application name, the setting applies to all applications and databases on the server.

- *dbname*—If you supply a database name, the setting applies only to the database. If you do not also provide 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)
 - 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

This setting specifies, in bytes, kilobytes, megabytes, or gigabytes, the maximum amount of memory that Essbase can allocate 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.

When the DYNCALCCACHEMAXSIZE setting is not equal to 0, you should also consider the following settings that affect how Essbase uses dynamic calculator cache:

- [“DYNCALCCACHEONLY” on page 435](#)
- [“DYNCALCCACHEWAITFORBLK” on page 436](#)
- [“DYNCALCCACHEBLKTIMEOUT” on page 430](#)
- [“DYNCALCCACHEBLKRELEASE” on page 429](#)
- [“DYNCALCCACHECOMPRBLKBUFSIZE” on page 432](#)

Notes

- Use [“DYNCALCCACHEWAITFORBLK” on page 436](#) and [“DYNCALCCACHEONLY” on page 435](#) to set or change how Essbase handles the situation when it has reached the maximum dynamic calculator cache size and needs more memory in the dynamic calculator cache to store dynamically calculated blocks.
- See the *Oracle Essbase Database Administrator's Guide* for more information about Dynamic Calculator Cache and the related configuration file settings.

Example

```
DYNCALCCACHEMAXSIZE 30M
```

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

See Also

[“DYNALCCACHEONLY” on page 435](#)

[“DYNALCCACHEWAITFORBLK” on page 436](#)

[“DYNALCCACHEBLKTIMEOUT” on page 430](#)

[“DYNALCCACHEBLKRELEASE” on page 429](#)

[“DYNALCCACHECOMPRBLKBUFSIZE” on page 432](#)

Oracle Essbase Database Administrator's Guide

DYNALCCACHEONLY

Specifies whether dynamic calculations can use memory outside the dynamic calculator cache in the case that it is full.

This setting does not apply to aggregate storage databases.

Syntax

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

- *appname*—If you supply an application name, the setting applies to all databases within the application. If you do not supply an application name, the setting applies to all applications and databases on the server.
- *dbname*—If you supply a database name, the setting applies only to the database. If you do not also provide an application name, the setting applies to all applications and databases on the server.
- TRUE—Disallows the use of memory outside the dynamic calculator cache. If space for blocks with dynamically calculated members cannot be obtained from the dynamic calculator cache, Essbase generates an error message.
- FALSE—Allows the use of memory outside the dynamic calculator cache, if necessary, for blocks containing dynamically calculated members. The default value is FALSE.

Description

When no room is available in the dynamic calculator cache, the [“DYNALCCACHEWAITFORBLK” on page 436](#) and [“DYNALCCACHECOMPRBLKBUFSIZE” on page 432](#) configuration settings provide options that could result in Essbase using memory outside the dynamic calculator cache to store blocks that contain dynamically calculated members. If you are experiencing a severe memory shortage, you can use the DYNALCCACHEONLY setting to disallow the use of memory outside the dynamic calculator cache. If DYNALCCACHEONLY is set to TRUE, instead of using memory outside the dynamic calculator cache, Essbase generates the error message, "Allocation outside the dynamic calculator cache is disallowed."

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 the dynamic calculator cache may improve retrieval performance by reducing the number of calls to the operating system to do memory allocations. The size of the improvement depends on your database configuration.

Notes

The default value of this setting is FALSE. Only set this value to TRUE for one or more of the following circumstances:

- The operating system is not properly reclaiming memory outside the dynamic calculator cache.
- There is a severe memory shortage
- Tighter control is required over memory usage for dynamic calculations

Example

```
DYNCALCCACHEONLY TRUE
```

Specifies that the dynamic calculator cache is the only memory area that Essbase may use to store blocks that contain dynamically calculated blocks. If a retrieval requires space that is not available in the dynamic calculator cache, the execution of the retrieval is aborted. The user sees an error message that is also posted to the application log.

See Also

[“DYNCALCCACHEMAXSIZE” on page 433](#)

[“DYNCALCCACHEWAITFORBLK” on page 436](#)

[“DYNCALCCACHEBLKTIMEOUT” on page 430](#)

[“DYNCALCCACHECOMPRBLKBUFSIZE” on page 432](#)

[“DYNCALCCACHEBLKRELEASE” on page 429](#)

DYNCALCCACHEWAITFORBLK

Specifies whether Essbase should wait for memory to be freed in the dynamic calculator cache, or use outside memory.

This setting does not apply to aggregate storage databases.

Syntax

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

- *appname*—If you supply an application name, the setting applies to all databases within the application. If you do not supply an application name, the setting applies to all applications and databases on the server.

- *dbname*—If you supply a database name, the setting applies only to the database. If you do not also provide an application name, the setting applies to all applications and databases on the server.
- TRUE—Tells Essbase to wait for memory to be freed in the dynamic calculator cache.
- FALSE—This is the default. If allowed by the [“DYNALCCACHEONLY” on page 435](#) setting, tells Essbase attempt to perform calculations on these blocks in memory outside the dynamic calculator cache.

If the [“DYNALCCACHEONLY” on page 435](#) setting is TRUE, tells Essbase to generate an error message instead of using memory outside the dynamic calculator cache.

Description

Use this setting to set or change how Essbase handles the situation when it needs additional memory to store blocks in the dynamic calculator cache for the database.

When the setting is TRUE, Essbase waits to store and calculate data blocks in the dynamic-calculator-cache area that is currently in use by other queries.

When the setting is FALSE, if the [“DYNALCCACHEONLY” on page 435](#) setting is also FALSE, instead of waiting for area in the dynamic calculator cache, Essbase attempts to store and calculate data blocks for the current query in memory outside the dynamic calculator cache. If the [“DYNALCCACHEONLY” on page 435](#) setting is TRUE, Essbase generates an error message instead of using memory outside the dynamic calculator cache.

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 the dynamic calculator cache may improve retrieval performance by reducing the number of calls to the operating system to do memory allocations. The size of the improvement depends on your database configuration.

Notes

Use the [“DYNALCCACHEBLKTIMEOUT” on page 430](#) setting to specify the maximum number of seconds that Essbase waits for space in the dynamic calculator cache.

Example

```
DYNALCCACHEONLY FALSE
DYNALCCACHEWAITFORBLK FALSE
```

Essbase attempts to perform the block calculation in memory outside the dynamic calculator cache, instead of waiting for space to become available in the dynamic calculator cache.

See Also

[“DYNALCCACHEMAXSIZE” on page 433](#)

[“DYNALCCACHEONLY” on page 435](#)

[“DYNCALCCACHEBLKTIMEOUT” on page 430](#)

[“DYNCALCCACHEBLKRELEASE” on page 429](#)

[“DYNCALCCACHECOMPRBLKBUFSIZE” on page 432](#)

ENABLE_DIAG_TRANSPARENT_PARTITION

Specifies whether to log transaction response times for requests sent from a data source to a transparent partition target. The target can be either a block storage or aggregate storage database. Logging these diagnostic messages is helpful when troubleshooting response times that are too slow.

Syntax

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

- *appname*—Optional. Specifies the application for which logging diagnostic messages is to be enabled.

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 logging diagnostic messages is to be enabled.

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.
- TRUE | FALSE—Specifies whether to enable or disable logging transaction response times for requests to a transparent partition.

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

Description

When logging is enabled, Essbase writes messages to the source and target database log files during querying.

For every partial response sent to the target from the source, Essbase logs these messages:

- In the source database log file, the following message, of type INFO, provides the size of the response grid:

Sending response grid of size xxxxx.
- In the target database log file, the following message provides the size of the request grid issued to the source and an estimated response time:

Waiting for data from source *system:application:database* grid size *sizeOfRequestGrid*. Approximately one second is needed to fetch a grid of size one million cells with non-missing cell density of 7% from the source.

For every partial grid received from the source, Essbase logs the following message about the density of the grid to the target database log file:

```
Density of the grid xxxxxx of fetch size xxxxxx.
```

When an aggregate storage database is the target of a transparent partition, you can set the request and response grid size.

Example

```
ENABLE_DIAG_TRANSPARENT_PARTITION ASOSamp TRUE
```

Enables logging of transaction response times for all databases associated with the ASOSamp application.

See Also

[“MAX_REQUEST_GRID_SIZE” on page 468](#) configuration setting

[“MAX_RESPONSE_GRID_SIZE” on page 469](#) configuration setting

ENABLECLEARMODE

Determines whether Essbase allows SSL connectivity.

Syntax

```
ENABLECLEARMODE TRUE | FALSE
```

- TRUE—Essbase handles plain TCP requests. The default value is TRUE.
- FALSE—Essbase handles only SSL requests, not plain TCP requests

Description

This setting determines whether Essbase allows -SSL connectivity. It applies only to Essbase Agent and applications.

Example

```
ENABLECLEARMODE FALSE
```

See Also

[“AGENTSECUREPORT” on page 388](#)

[“CLIENTPREFERREDMODE” on page 411](#)

[“ENABLESECUREMODE” on page 440](#)

[“NETSSLHANDSHAKETIMEOUT” on page 474](#)

[“SSLCIPHERSUITES” on page 502](#)

[“WALLETPATH” on page 515](#)

For information on implementing SSL, see the *Oracle Hyperion Enterprise Performance Management System Security Administration Guide*.

ENABLESECUREMODE

Allows -Secure Socket Layer (SSL) connectivity to Essbase.

Syntax

```
ENABLESECUREMODE TRUE | FALSE
```

- TRUE—SSL is enabled. Essbase can handle SSL requests.
- FALSE—SSL is not loaded and not used. The default value is FALSE.

Description

This setting determines whether Essbase allows -SSL connectivity. It applies only to Essbase Agent and applications.

Example

```
ENABLESECUREMODE TRUE
```

See Also

[“AGENTSECUREPORT” on page 388](#)

[“CLIENTPREFERREDMODE” on page 411](#)

[“ENABLECLEARMODE” on page 439](#)

[“NETSSLHANDSHAKETIMEOUT” on page 474](#)

[“SSLCIPHERSUITES” on page 502](#)

[“WALLETPATH” on page 515](#)

For information on implementing SSL, see the *Oracle Hyperion Enterprise Performance Management System Security Administration Guide*.

ENABLESWITCHTOBACKUPFILE

Specifies whether to load the latest, valid backup security file (`essbase_timestamp.bak`) at startup if the `essbase.sec` file is invalid.

Syntax

```
ENABLESWITCHTOBACKUPFILE TRUE | FALSE
```


- TRUE—If `essbase.sec` is invalid at startup, Essbase cycles through the `essbase_timestamp.bak` files, starting with the backup file with the latest timestamp, until it finds a valid backup file with which to start Essbase.
- FALSE—If `essbase.sec` is invalid, Essbase startup is aborted and a message is written to the `essbase.log` file. The Essbase administrator must restore `essbase.sec` by copying the latest, valid backup file to it.

The default value is FALSE.

Note: You can configure the number of backup security files that Essbase creates and maintains, and the interval in which Essbase creates backup security files.

Example

```
ENABLESWITCHTOBACKUPFILE TRUE
```

See Also

[“NUMBEROFSECFILEBACKUPS” on page 476](#)

[“SECFILEBACKUPINTERVAL” on page 490](#)

ESSBASEFAILOVERTRACELEVEL

Sets the trace level for messages written to the Lease Manager log files.

Syntax

```
ESSBASEFAILOVERTRACELEVEL USER | ADMIN
```

Where USER and ADMIN are priority levels:

- USER—Lease renewal messages are written to the log files whenever a new lease is acquired. Lease ownership messages are not written to the log files. This is the default setting.
- ADMIN—Lease renewal and lease ownership messages are written to the log files every time a lease is renewed.

Example

```
ESSBASEFAILOVERTRACELEVEL ADMIN
```

Sets the trace level to ADMIN which writes the messages Lease manager has a current lease and Lease Manager successfully acquired/renewed its lease to the Lease Manager log files every time a lease is renewed.

See Also

[“FAILOVERMODE” on page 445](#)

ESSBASESERVERHOSTNAME

Specifies the computer host name to which Essbase Agent and Essbase Server bind and where an Essbase application process runs.

Syntax

```
ESSBASESERVERHOSTNAME server_name
```

Where *host_name* is the name of the host where your Essbase application process runs. ESSBASESERVERHOSTNAME uses the current server by default.

Description

ESSBASESERVERHOSTNAME identifies the host where your Essbase application process runs. The value must be a valid host name and must map to an IP address assigned to the computer. If ESSBASESERVERHOSTNAME is not specified in `essbase.cfg`, Essbase and the applications listen on all interfaces (IP_ANY).

Notes

- You can use ESSBASESERVERHOSTNAME to restrict the network interface provide on which Essbase and the applications listen when multiple instances are running on the same host computer.
- For information on running multiple Essbase instances on a single computer, see:
 - [“AGENTPORT” on page 387](#)
 - [“SERVERPORTBEGIN” on page 493](#)
 - [“SERVERPORTEND” on page 494](#)
 - [“PORTINC” on page 481](#)
- The behavior of the client is not necessarily tied to this configuration setting. For example, the MaxL client always uses localhost as the default, irrespective of this configuration setting.
- In Report Writer, you can display ESSBASESERVERHOSTNAME values on a report. For example, you can use the *MACHINE replacement value in the Report Writer {Mask} command to display the ESSBASESERVERHOSTNAME value as the server name.

Example

```
ESSBASESERVERHOSTNAME Hyper
```

identifies the host name "Hyper".

EXCEPTIONLOGOVERWRITE

Determines whether Essbase overwrites the existing exception log or creates a new exception log.

Syntax

EXCEPTIONLOGOVERWRITE TRUE | FALSE

- TRUE—Essbase overwrites the existing exception log.
- FALSE—Essbase keeps the existing exception log and creates new logs for every exception. The default value is FALSE.

Description

This setting determines whether Essbase overwrites existing exception log data or creates a new log for each exception condition. The exception log name is normally `log00001.xcp`.

When `EXCEPTIONLOGOVERWRITE` is FALSE:

- Essbase creates a new log instead of overwriting the previous one.
- Subsequent logs are numbered sequentially; for example, if `log00001.xcp` exists, the next log has the file name `log00002.xcp`, and the next has `log00003.xcp`, and so on.

The Essbase exception handler writes the information into the exception log on the local disk in a text file as follows:

- If the server crashed, the log is written in the directory pointed to by `ESSBASEPATH`; for example, `D:\essbase`
- If the application crashed and the application name is unknown, the log is written into the `APP` subdirectory under the directory pointed to by `ARBORPATH`; for example, `D:\essbase\app`.
- If the application crashed and the application name is known, but the database name is unknown, the log is written to the appropriate application directory; for example, `D:\essbase\app\app1`.
- If the application crashed and both the application and database names are known, the log is written to the appropriate database directory; for example, `D:\essbase\app\app1\db1`.

Notes

- When an exception occurs, Essbase displays and logs an error message telling users the path to the exception log.
- Essbase logs errors to the Essbase Server log or to the application log, depending on where the error occurs.

Example

```
EXCEPTIONLOGOVERWRITE FALSE
```

See Also

Oracle Essbase Database Administrator's Guide

EXCLUSIVECALC

Determines whether Essbase allows concurrent calculations.

This setting does not apply to aggregate storage databases.

Syntax

```
EXCLUSIVECALC TRUE | FALSE
```

- **TRUE**—If a calculation operation (command or script) is running, Essbase fails any other calculation operations.
- **FALSE**—This is the default. Essbase allows concurrent calculation operations.

Description

This setting determines whether Essbase runs calculations concurrently in the same database. Essbase prevents any other calculation operations from executing on the same database.

Example

```
EXCLUSIVECALC TRUE
```

EXPORTTHREADS

Sets the the default number of threads that can be produced during parallel data export.

Syntax

```
EXPORTTHREADS appname dbname n
```

- *appname*—This is the name of the application. You can also use `xxxxxx` as a wildcard to indicate all application names.
- *dbname*—This is the name of the database. You can also use `xxxxxx` as a wildcard to indicate all database names.
- *n*—This integer, between 1 and 8, inclusive, sets the default for the number of export threads that can be used to export data. This number should generally be equal to the number of processors on your machine that you wish to commit to doing parallel export. The default is 1.

Description

This setting enables the user to specify the number of threads that can be used to export data. The export process is then executed in parallel, and multiple threads can retrieve data and write to their corresponding export files concurrently. If `EXPORTTHREADS` is not specified, or is not followed by its arguments, then the default value of 1 is used.

Notes

For more information about the export utility, see the *Oracle Essbase Database Administrator's Guide*.

Example

```
EXPORTTHREADS sample basic 4
```

See Also

[Export Data \(MaxL\)](#)

[PAREXPORT \(ESSCMD\)](#)

FAILOVERMODE

Determines whether Essbase is deployed as a failover cluster.

Syntax

```
FAILOVERMODE TRUE | FALSE
```

- TRUE—Essbase runs as a failover cluster managed by Oracle Process Manager and Notification Server.

Note that on UNIX systems, enabling FAILOVERMODE sets [FILELOCKINGMODE](#) to NONE.

- FALSE—Essbase runs as a stand-alone server. The default value is FALSE.

Description

This setting determines whether Essbase is deployed as a failover cluster that is managed by OPMN, or as a stand-alone server.

When FAILOVERMODE is TRUE:

- The Essbase cluster must be started and stopped using OPMN.
- The `opmn.xml` file must be modified to ensure that OPMN is aware of the Essbase cluster.

Example

```
FAILOVERMODE FALSE
```

See Also

Oracle Essbase Database Administrator's Guide

[“AGENTLEASEEXPIRATIONTIME” on page 384](#)

[“AGENTLEASEMAXRETRYCOUNT” on page 385](#)

[“AGENTLEASERENEWALTIME” on page 385](#)

[“SERVERLEASEEXPIRATIONTIME” on page 492](#)

[“SERVERLEASEMAXRETRYCOUNT” on page 492](#)

[“SERVERLEASERENEWALTIME” on page 493](#)

FILELOCKINGMODE

On UNIX, provides a way for the operating system to limit file access to only one process (user).

Note: This setting does not apply to Windows systems.

Syntax

FILELOCKINGMODE Advisory | Mandatory | None

- **Advisory**—Locks files. All applications that follow rules (like Essbase) will honor the locks. This is the default setting.

Note: When [FAILOVERMODE](#) is set to TRUE, setting FILELOCKINGMODE to Advisory or Mandatory has no effect.

- **Mandatory**—Locks files at the kernel level. This setting provides extra security to protect against malicious software.
- **None**—No files are locked. Two Essbase instances can modify the same data and potentially corrupt it. This option is added for Failover mode where file access is not done by the operating system, but by an application acquiring a lease (use database).

Description

FILELOCKINGMODE specifies how files are locked on UNIX systems.

Example

```
FILELOCKINGMODE Advisory
```

See Also

[FAILOVERMODE](#)

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
```

FORCEGRIDEXPANSION

When set to ON, forces the expansion of the grid when transparent partitions are queried, thus ensuring that correct results are retrieved when most data values are displayed as #MISSING, whether or not cells contain data.

The FORCEGRIDEXPANSION configuration setting is used with the GRIDEXPANSION configuration setting.

Syntax

```
FORCEGRIDEXPANSION ON | OFF
```

The default value is OFF.

Description

If GRIDEXPANSION is set to ON, the grid is not expanded if all of the following conditions are met, and, thus, incorrect results are returned:

- 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.

When both GRIDEXPANSION and FORCEGRIDEXPANSION are set to ON, the grid is expanded and the correct values for cells that contain data are displayed. Query performance, however, is slowed.

If GRIDEXPANSION is set to OFF, the FORCEGRIDEXPANSION setting is ignored.

See Also

[“GRIDEXPANSION” on page 448](#)

[“GRIDEXPANSIONMESSAGES” on page 448](#)

GRIDEXPANSION

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

Syntax

GRIDEXPANSION ON | OFF

The default value is ON.

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.

If client queries receive incorrect results, set FORCEGRIDEXPANSION to ON. (If GRIDEXPANSION is set to OFF, the FORCEGRIDEXPANSION setting is ignored.)

See Also

[“FORCEGRIDEXPANSION” on page 447](#)

[“GRIDEXPANSIONMESSAGES” on page 448](#)

GRIDEXPANSIONMESSAGES

Sets whether grid expansion-related messages are displayed to Spreadsheet Add-in users and written to the application log.

Syntax

GRIDEXPANSIONMESSAGES ON | OFF

- ON—Allows grid-expansion-related messages.
- OFF—This is the default value. Suppresses grid-expansion-related messages.

Description

If a Spreadsheet Add-in 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” on page 448](#)

[“FORCEGRIDEXPANSION” on page 447](#)

HAENABLE

Sets whether members can be retrieved from a Hybrid Analysis relational source.

Syntax

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

- *appname*—Optional. If you supply an application name, the setting applies to all databases within the named application. If you do not supply an application name, the setting applies to all applications and databases on the Essbase Server.
- *dbname*—Optional. If you supply a database name and an application name, the setting applies only to the named database. If you do not also provide an application name, the database is ignored and the setting applies to all applications and databases on the Essbase Server.
- TRUE—Essbase retrieves all members of a Hybrid Analysis Relational Source through API's, reports, or Spreadsheet Add-in. If HAENABLE is on, requests can transparently span across the Hybrid Analysis Relational Source.
- FALSE—This is the default. Essbase turns off retrieval of members of a Hybrid Analysis Relational Source for all clients.

Description

This setting globally turns on or off the ability to retrieve members of a Hybrid Analysis Relational Source.

Example

```
HAENABLE FALSE
```

HAMAXNUMCONNECTION

Sets the maximum number of connections per database that Essbase can keep active against the relational database.

Syntax

```
HAMAXNUMCONNECTION [appname [dbname]] n
```

- *appname*—Optional. If you supply an application name, the setting applies to all databases within the named application. If you do not supply an application name, the setting applies to all applications and databases on the Essbase Server.
- *dbname*—Optional. If you supply a database name and an application name, the setting applies only to the named database. If you do not also provide an application name, the database is ignored and the setting applies to all applications and databases on the Essbase Server.
- *n*—Specifies the number of connections per database that Essbase can keep connected to a relational database. The default is 25.

Description

This setting determines the maximum number of connections per database that Essbase can keep active against the relational database. This optimizes the overhead involved in opening a relational database connection for every Hybrid Analysis report.

Notes

- You may need to set the value for the HAMAXNUMCONNECTION higher if, in Integration Services Console, you set the security mode to use the Essbase user ID to connect to the source database. Refer to the *Oracle Essbase Integration Services Online Help* for more information on the security mode setting.
- This setting is interpreted as 0 (zero) for Advanced Relational Analysis (ARA) because ARA does not support a connection pool.

Example

```
HAMAXNUMCONNECTION 10
```

HAMAXNUMSQLQUERY

Sets the maximum number of SQL queries that can be issued against the fact table(s) in the relational database per Essbase query session.

Syntax

```
HAMAXNUMSQLQUERY [appname [dbname]] n
```

- *appname*—Optional. If you supply an application name, the setting applies to all databases within the named application. If you do not supply an application name, the setting applies to all applications and databases on the Essbase Server.
- *dbname*—Optional. If you supply a database name and an application name, the setting applies only to the named database. If you do not also provide an application name, the database is ignored and the setting applies to all applications and databases on the Essbase Server.
- *n*—The value of *n* is the number of simultaneous SQL queries per Essbase query session. The default is 50.

Description

This setting determines the maximum number of SQL queries that can be executed during an Essbase query session from Report Writer or from the Spreadsheet Add-in extractor. If a query cannot be split into pieces in such a way as to not violate the limits set by this command, the query execution fails. The user has to submit a smaller version of the query, or the administrator must raise the value of this setting. HAMAXNUMSQLQUERY does not refer to the number of users performing queries; rather, it refers to the number of SQL queries in a complex statement.

Example

In the following example, the maximum number of SQL queries per Essbase query session is set to 10.

```
HAMAXNUMSQLQUERY 10
```

See Also

[“HAMAXQUERYROWS” on page 451](#)

[“HARETRIEVENUMROW” on page 455](#)

[“QRYGOVEXECBLK” on page 485](#)

[“QRYGOVEXECTIME” on page 486](#)

HAMAXQUERYROWS

Sets the maximum number of rows that can be returned per SQL query issued on behalf of an Essbase query.

Syntax

```
HAMAXQUERYROWS [appname [dbname]] n
```

- *appname*—Optional. If you supply an application name, the setting applies to all databases within the named application. If you do not supply an application name, the setting applies to all applications and databases on the Essbase Server.
- *dbname*—Optional. If you supply a database name and an application name, the setting applies only to the named database. If you do not also provide an application name, the database is ignored and the setting applies to all applications and databases on the Essbase Server.
- *n*—Determines the maximum number of rows retrieved per SQL query. The default is zero, meaning that no row limit is applied.

Description

This setting determines the maximum number of rows retrieved per SQL query issued on behalf of an Essbase query. HAMAXQUERYROWS is a specific limit to an Essbase query. The setting provides a way of controlling queries that retrieve too much data.

Notes

- An important distinction exists between the purposes of HAMAXQUERYROWS and HARETRIEVENUMROW. Whereas HAMAXQUERYROWS controls the number of total rows to return, HARETRIEVENUMROW affects memory consumption by controlling how many rows to process at one time.
- HAMAXQUERYROWS and HARETRIEVENUMROW should not be confused with QRYGOVEXECBLK which sets the maximum number of *blocks* that a query can access before the query is terminated.

Example

In the following example, Essbase processes up to 100,000 rows per SQL query.

```
HAMAXQUERYROWS 100000
```

See Also

[“HAMAXNUMSQLQUERY” on page 450](#)

[“HARETRIEVENUMROW” on page 455](#)

[“QRYGOVEXECBLK” on page 485](#)

[“QRYGOVEXECTIME” on page 486](#)

HAMAXQUERYTIME

Sets the maximum time limit per query for SQL queries from a Hybrid Analysis Relational Source. When set on a Oracle datasource, this setting is ignored and a warning message is logged in the application log file.

Syntax

```
HAMAXQUERYTIME [appname [dbname]] n
```

- *appname*—Optional. If you supply an application name, the setting applies to all databases within the named application. If you do not supply an application name, the setting applies to all applications and databases on the Essbase Server.
- *dbname*—Optional. If you supply a database name and an application name, the setting applies only to the named database. If you do not also provide an application name, the database is ignored and the setting applies to all applications and databases on the Essbase Server.
- *n*—Determines the time limit per query in seconds. The default is zero, meaning that no time limit is applied.

Description

This setting determines how much time an SQL query operation can take before it is forcefully terminated. HAMAXQUERYTIME is a specific limit to an Essbase query that spans the Hybrid

Analysis Relational Source and is set in the `essbase.cfg` file. The value set is dependent on how long an SQL query issued on behalf of an Essbase query can take to complete.

An important distinction exists between the purposes of `HAMAXQUERYTIME` and `QRYGOVEXEETIME`. Note that `QRYGOVEXEETIME` affects the entire query, such as from the time a user double-clicks a cell to retrieve data in Spreadsheet Add-in to the time the results are displayed. `HAMAXQUERYTIME`, on the other hand, affects only a portion of the Essbase query, such as the individual SQL queries from the Hybrid Analysis Relational Source. When a query spans Hybrid Analysis data, `QRYGOVEXEETIME` is disabled for the rest of the overall query, and the query timer controlled by `HAMAXQUERYTIME` takes effect.

Example

```
HAMAXQUERYTIME 300
```

See Also

[“HAMAXNUMSQLQUERY” on page 450](#)

[“HARETRIEVENUMROW” on page 455](#)

[“QRYGOVEXECBLK” on page 485](#)

[“QRYGOVEXEETIME” on page 486](#)

HAMEMORYCACHESIZE

Determines the amount of memory that is reserved to cache queried members from a Hybrid Analysis Relational Source.

Syntax

```
HAMEMORYCACHESIZE [appname [dbname]] n
```

- *appname*—Optional. If you supply an application name, the setting applies to all databases within the named application. If you do not supply an application name, the setting applies to all applications and databases on the Essbase Server.
- *dbname*—Optional. If you supply a database name and an application name, the setting applies only to the named database. If you do not also provide an application name, the database is ignored and the setting applies to all applications and databases on the Essbase Server.
- *n*—An integer expressed in bytes (B), kilobytes (K), megabytes (M), or gigabytes (G)
 - Minimum value: 1048576 B (1 M). Any value less than this minimum value causes `HAMEMORYCACHESIZE` to default to 1 M.
 - Default value: 1 megabytes (1 M, which is 1048576 bytes)
 - 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

This setting sizes the memory buffer that holds relational data during the execution of spread sheet or report scripts that drill into Hybrid Analysis Relational Sources. When you specify the cache size, you control the memory used to cache relational members during execution. A larger cache size optimizes the usage of relational members during execution and increases the speed of metadata retrieval from the metaoutline. Thus, more memory allocated to the cache means fewer SQL queries to resolve member names, resulting in improved performance.

Notes

In order to have configuration parameters applied to the application level, the global value must be defined first, and the application level value must be defined second (see Example 2 below).

Examples

Example 1

```
HAMEMORYCACHESIZE 1M
```

Example 2

In the following example, all databases in the application "Test" have a cache value of 1500K, and all other applications have a cache value of 2M.

```
HAMEMORYCACHESIZE 2M  
HAMEMORYCACHESIZE Test 1500K
```

In the following example, all applications and databases have a cache value of 2M because the application cache value is overridden.

```
HAMEMORYCACHESIZE Test 1500K  
HAMEMORYCACHESIZE 2M
```

This ordering is needed only for global and unspecified (untitled) applications.

HARAGGEDHIERARCHY

Enables support of null values in columns of dimension tables that are used to create dimensions for Hybrid Analysis-enabled outlines.

Syntax

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

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

- TRUE—Setting the value to TRUE enables Hybrid Analysis to account for null values during SQL generation. Note that this setting may have impact on performance because SQL generation is not optimized well on most relational databases.
- FALSE—Setting the value to FALSE prevents Hybrid Analysis from recognizing the null values or gaps in the ragged hierarchy. This option does not affect performance. The default is FALSE.

Description

This setting enables support of null values in columns of dimension tables that are used to create dimensions for Hybrid Analysis-enabled outlines. The dimension build that these dimension tables result in the outlines are known as ragged hierarchies.

If you have null values in columns of dimension tables used to create dimensions in a Hybrid Analysis-enabled outline, use the HARAGGEDHIERARCHY setting in `essbase.cfg` to enable Hybrid Analysis to account for these null values during SQL generation.

Example

In the following example, TRUE enables Hybrid Analysis to account for null values during SQL generation.

```
HARAGGEDHIERARCHY TRUE
```

HARETRIEVENUMROW

Sets the maximum number of rows resulting from an SQL query to process at one time.

Syntax

```
HARETRIEVENUMROW [appname [dbname]] n
```

- *appname*—Optional. If you supply an application name, the setting applies to all databases within the named application. If you do not supply an application name, the setting applies to all applications and databases on the Essbase Server.
- *dbname*—Optional. If you supply a database name and an application name, the setting applies only to the named database. If you do not also provide an application name, the database is ignored and the setting applies to all applications and databases on the Essbase Server.
- *n*—The value of *n* specifies how many rows to process at a time. The default is 100.

Description

This setting sets the number of rows to process at a time. A low value may degrade performance and increase query time, but reduce memory usage.

Notes

An important distinction exists between the purposes of HARETRIEVENUMROW and HAMAXQUERYROWS. Whereas HARETRIEVENUMROW affects memory consumption by

controlling how many rows to process at one time, HAMAXQUERYROWS controls the number of total rows to return.

HARETRIEVENUMROW and HAMAXQUERYROWS should not be confused with QRYGOVEXECBLK which sets the maximum number of *blocks* that a query can access before the query is terminated.

Example

In the following example, an Essbase query processes rows from each SQL query in sets of 50 rows.

```
HARETRIEVENUMROW 50
```

See Also

[“HAMAXNUMSQLQUERY” on page 450](#)

[“HAMAXQUERYROWS” on page 451](#)

HASOURCEDSNOS390

Set to TRUE if the DB2 data source for Hybrid Analysis resides on an OS/390 system.

Syntax

```
HASOURCEDSNOS390 appname dbname TRUE | FALSE
```

- *appname*—The name of the application affected by this setting. You can also use `xxxxxx` as a wildcard to indicate all applications.
- *dbname*—The name of the database affected by this setting. You can also use `xxxxxx` as a wildcard to indicate all databases in the specified application (or all databases on the server, if `xxxxxx` is used in place of *appname*).
- TRUE—Essbase recognizes a DB2 database as a Hybrid Analysis source on an OS/390 system. DB2 databases on other operating systems are not recognized.
- FALSE—Reverses the effect of the TRUE setting. This is the default state.

Examples

```
HASOURCEDSNOS390 xxxxxx xxxxxx TRUE
```

For all databases on the server, Essbase recognizes a DB2 database as a Hybrid Analysis source on an OS/390 system.

```
HASOURCEDSNOS390 Sample xxxxxx TRUE
```

For all databases in the Sample application, Essbase recognizes a DB2 database as a Hybrid Analysis source on an OS/390 system.

```
HASOURCEDSNOS390 Sample Basic TRUE
```

For the Basic database in the Sample application, Essbase recognizes a DB2 database as a Hybrid Analysis source on an OS/390 system.

HISLEVELDRILLTHROUGH

For an intersection to be available in a drill-through report, specifies that for each member in the intersection the generation must be equal to or greater than the generation defined in the report and the level must be equal to or lesser than the level defined in the report.

When HISLEVELDRILLTHROUGH is set, intersections whose members do not meet this criteria are not available for drill-through. For example, when a member is promoted in the hierarchy while creating an outline, the parent in the source database becomes null and the hierarchy becomes ragged. The intersection that contains the null parent is excluded in the drill-through report.

Syntax

```
HISLEVELDRILLTHROUGH appname
```

- *appname*—Specifies the application for which intersections must be well formed to be available in drill-through reports.

You must restart the application to initialize any change to the configuration file.

Example

```
HISLEVELDRILLTHROUGH Sample
```

Specifies that, in the Sample application, intersections must be well formed to be available in drill-through reports.

IBHFIXTHRESHOLD

Controls how many invalid block-header messages are returned to the client or server log, relative to the number of level-0 blocks written to disk.

This setting does not apply to aggregate storage databases.

Syntax

```
IBHFIXTHRESHOLD appName | xxxx dbName | xxxx percentage
```

- *appName*—Optional. If you supply an application name but use *xxxx* in place of *dbName*, the setting applies to all databases within the named application. If you use *xxxx* in place of *appName* and *dbName*, the setting applies to all applications and databases on the server.
- *dbName*—Optional. If you supply a database name and an application name, the setting applies only to the named database. If you use *xxxx* in place of *dbName*, the setting applies to all databases within the named application. If you supply a database name, you must also supply an application name.
- *xxxx*—If used in place of *appName*, and *dbName*, specifies all databases on the Essbase Server. If used in place of *dbName*, specifies all databases on the application.
- *percentage*—Percentage of invalid block-header errors to report, relative to the number of level-0 blocks on disk. Once the threshold is reached, a message is sent to the client requesting

that the user rebuild the database, and the Essbase Server shuts down. Valid values are integers 0 to 100.

Description

You must set the server configuration setting IBHFIXTHRESHOLD in the server `essbase.cfg` file and restart Essbase Server before you can find and fix invalid block-header problems.

This setting controls how many invalid block-header messages are returned to the client or server log, relative to the number of level 0 blocks written to disk. After the threshold is reached, no corrective action can be performed, and a message is sent to the client suggesting that the database be rebuilt.

If messages are written to the client or server log indicating the presence of invalid block-header errors, but the threshold that requires the database be rebuilt is not reached, you can either rebuild the database or you can find and fix the errors using MaxL: [Alter Database](#) DBS-NAME validate data to local | server logfile FILE-NAME.

Notes

- If Essbase runs in uncommitted mode when it receives an invalid block-header error message, the current transaction may stop without any rollback, meaning that some data may have changed. Be sure to verify that all transactions that you expected to finish have finished. If not, you may need to clean up the data or rebuild the database.
- For information about types of invalid block-header errors and how to rebuild a database, see the *Oracle Essbase Database Administrator's Guide*.

Example

```
IBHFIXTHRESHOLD sample basic 10
```

Specifies that on Sample Basic, if 10% of the data blocks have invalid block-header errors, it is time to rebuild the database.

```
IBHFIXTHRESHOLD sample xxxx 15
```

Specifies that for any database in the Sample application, if 15% of the data blocks have invalid block-header errors, it is time to rebuild the database.

```
IBHFIXTHRESHOLD xxxx xxxx 5
```

Specifies that for any database, if 5% of the data blocks have invalid block-header errors, it is time to rebuild the database.

See Also

[Alter Database](#) (MaxL statement)

IDMIGRATION

Controls whether the unique identifiers from Shared Services are added to Essbase user and group IDs.

Syntax

```
IDMIGRATION CHECKANDMIGRATE | NOMIGRATION | FORCEMIGRATION
```

- **CHECKANDMIGRATE**—Default option. Checks for identity attributes that have changed in Shared Services and updates them in Essbase security.
- **NOMIGRATION**—Makes no changes in Essbase security.
- **FORCEMIGRATION**—Updates Essbase users and groups without checking whether identity attributes have changed.

Description

With release 9.2.0.3, a unique identity field was added to user and group IDs to ensure the IDs across Shared Services and Essbase could be uniquely identified. By default, after installing release 9.2.0.3 (or 9.3.1 if release 9.2.0.3 was skipped), when Essbase Server is started it migrates changed Shared Services user and group IDs to include the unique identity field. You can use the **IDMIGRATION** configuration setting to skip this migration or to force migration of all user and group IDs.

Example

```
IDMIGRATION NOMIGRATION
```

IMPLIED_SHARE

Sets the default implied shared behavior for the Essbase Server, or for the specified application.

Syntax

```
IMPLIED_SHARE [app_name] TRUE | FALSE
```

- *app_name*—Optional. If provided, the setting applies only to the specified application; otherwise, the setting applies to the Essbase Server.
- **TRUE**—Default value. Parent is treated as an implied share because it has only one child or because it has only one child that consolidates to the parent.
- **FALSE**—Never use Implied Share unless explicitly set in the outline.

Notes

- If the **IMPLIED_SHARE** configuration setting is absent from `Essbase.cfg`, the default setting of **TRUE** is used.
- Application-specific settings overrides any general Essbase Server settings.
- Outlines settings (specified using the Outline Editor or through the API) override all `Essbase.cfg` file settings.

- You must stop and restart the Essbase Server for it to read any changes to `IMPLIED_SHARE` settings made in the configuration file.
- The outline must be opened, saved, and then restructured for any changes in the `IMPLIED_SHARE` configuration setting now read by the Essbase Server to take effect.
- For `IMPLIED_SHARE` configuration settings to have any effect, the outline-specific setting must be one of these default values:
 - `ESS_IMPLIEDSHARE_DEFAULT`—Outline setting depends on `Essbase.cfg`. If returned, no `IMPLIED_SHARE` entry existed in `Essbase.cfg` at the time the outline was saved.
 - `ESS_IMPLIEDSHARE_DEFAULT_ON`—Outline setting depends on `Essbase.cfg`. If returned, `IMPLIED_SHARE` was set to `TRUE` in `Essbase.cfg` at the time the outline was saved.
 - `ESS_IMPLIEDSHARE_DEFAULT_OFF`—Outline setting depends on `Essbase.cfg`. If returned, `IMPLIED_SHARE` was set to `FALSE` in `Essbase.cfg` at the time the outline was saved.

Example

```
IMPLIED_SHARE Sample FALSE
```

Never use Implied Share for application Sample unless it is explicitly set.

INCRESTRUC

Specifies whether incremental restructuring is enabled for a database. You can enable incremental restructuring for individual databases or for all databases.

This setting does not apply to aggregate storage databases.

Syntax

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

- *appname*—Application name. Optional parameter for enabling incremental restructuring for one or all databases in an application. This parameter may be used in combination with *dbname*. If you omit *appname*, you cannot specify *dbname*, and `INCRESTRUC` will be enabled for all applications and databases. See Example below.
- *dbname*—Database name. Optional parameter for enabling incremental restructuring for an individual database. This parameter must be used in combination with *appname*. If you specify *dbname*, you must also specify *appname*. See Example below.
- `TRUE`—When you make certain outline or dimension changes that normally result in immediate database restructuring, Essbase defers restructuring until the next time it accesses the affected blocks. See Notes below.
- `FALSE`—Essbase immediately restructures the database whenever an outline or dimension change calls for it. The default value is `FALSE` (for all databases).

Notes

- Use the value `xxxxxx` to indicate "all" for any application or database argument. For example: `INCRESTRUC xxxxxx Basic TRUE` enables incremental restructuring for any application with a Basic database.
- Settings for nonexistent applications or databases are ignored.
- You can issue up to ten (total) `INCRESTRUC` statements per application.

Description

This setting specifies whether incremental restructuring is enabled for a database. You can enable incremental restructuring for individual databases, for all databases in an application, or for all databases on a server.

If you make certain outline or dimension changes that normally result in immediate database restructuring, Essbase defers restructuring until the next time the affected block is accessed, or until a full restructure is forced (e.g., by a full calculation). For example, if you add a member to any dimension, or delete a member from a dense dimension, Essbase defers restructuring when you enable `INCRESTRUC`.

When incremental restructuring is enabled, Essbase defers restructuring if you change the database outline or a dimension in a way that does not cause structural changes.

The following changes result in incremental (deferred) restructuring:

- Adding a member to a sparse or dense dimension
- Deleting a member from a dense dimension
- Moving a member within a dense dimension
- Adding, moving, or deleting a Dynamic Calc member
- Adding, moving, or deleting a Dynamic Calc and Store member in a *dense dimension*
- Adding a Dynamic Calc and Store member in a *sparse dimension*
- Re-defining a Dynamic Calc member as type Dynamic Calc and Store
- Re-defining a Dynamic Calc and Store member as type Dynamic Calc
- Re-defining a Dynamic Calc or Dynamic Calc and Store member as a regular member
- Re-defining a regular member as type Dynamic Calc or Dynamic Calc and Store

Restructuring for Dynamic Calc members is different from restructuring for Dynamic Calc and Store members. In general, Dynamic Calc and Store members have a greater impact on restructuring.

The following changes result in immediate restructuring, regardless of whether incremental restructuring is enabled:

- Adding or deleting a dimension
- Deleting a stored member of a sparse dimension
- Moving a member in a sparse dimension
- Moving or deleting a Dynamic Calc and Store member in a *sparse dimension*

- Changing dimension definition from sparse to dense, or from dense to sparse
- Changing the order of sparse dimensions
 - Certain member additions or changes to sparse dimensions can also trigger immediate restructuring.
- Changing the order of dense dimensions

If an incremental restructure has already occurred and shared members are added to the outline, Essbase ignores the `INCRESTRUC` setting and performs a full restructure.

Essbase logs outline changes in an internal file, `database_name.oc1`. Essbase clears the file whenever it does a full database restructure or when you clear or reset a database.

The `database_name.oc1` file can grow quite large in the meantime. To clear this file, issue `VALIDATE` in `ESSCMD`. `VALIDATE` causes Essbase to restructure any blocks whose restructure was deferred, and clears the file. When you issue `VALIDATE`, make sure the database is not in Read-only mode (Read-only mode is used for archiving).

If set to `TRUE`, `INCRESTRUC` affects all databases in all applications on the Essbase Server (except databases containing LROs), unless you have specified an *appname* and *dbname*.

The settings for `INCRESTRUC` are applied according to their order of appearance in the `essbase.cfg` file. For example:

```
INCRESTRUC TRUE
INCRESTRUC Sample Basic FALSE
```

enables incremental restructuring for all databases *except* Sample Basic.

If you are using Linked Reporting Objects (LROs) in a database, incremental restructuring is automatically disabled on that database. When you have incremental restructuring enabled for all databases in all applications (that is, you have set `INCRESTRUC` to `TRUE`), the presence of an LRO in a database disables incremental restructuring for that database, but does not affect the other databases on the Essbase Server.

If you add shared members to an outline, incremental restructuring is automatically turned off. If a restructure is triggered by outline changes, it will be done.

For more information about incremental restructuring, see the *Oracle Essbase Database Administrator's Guide*.

Examples

```
INCRESTRUC Sample Basic TRUE
```

Defers restructuring the Basic database in the Sample application, whenever certain outline or dimension changes are made, until the next time Essbase accesses the affected blocks; that is, it enables incremental restructuring for that database.

```
INCRESTRUC Sample TRUE
```

Defers restructuring for all databases in the Sample application, whenever certain outline or dimension changes are made, until the next time Essbase accesses the affected blocks; that is, it enables incremental restructuring for those databases.

INCRESTRUC TRUE

Defers restructuring all databases, whenever certain outline or dimension changes are made, until the next time Essbase accesses the affected blocks; that is, it enables incremental restructuring for all databases in all applications on that server.

INCRESTRUC FALSE

Immediately restructures all databases whenever an outline or dimension change calls for it; that is, it disables incremental restructuring for all databases in all applications on that server.

INDEXCACHESIZE

Defines the initial value for the index cache size for any new databases that are created after Essbase is restarted. The index cache is a buffer in memory that holds index pages. Essbase allocates this memory at startup of the database.

This setting does not apply to aggregate storage databases.

Syntax

INDEXCACHESIZE *n*

Where *n* is an integer expressed in bytes (B), kilobytes (K), megabytes (M), or gigabytes (G):

- Minimum value: 1 megabyte (1 M)
- Maximum value: 2 gigabytes (2 G)
- Default value: 10 megabytes (10 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

This setting specifies, in bytes, kilobytes, megabytes, or gigabytes, the initial size of the index cache for newly created or migrated databases on the server. The specified value takes effect for all new databases that are created after the server is started. To set or change the index cache size for an individual database, use Administration Services or MaxL. For more information, see the online help or HTML documentation for those components.

Example

INDEXCACHESIZE 100M

sets the index cache size of all subsequently created databases at 100 megabytes.

JVMMODULELOCATION

Specifies a Java Virtual Machine (JVM) library to be used by Essbase. This parameter is useful if you have more than one version of the JVM library installed on the computer running Essbase.

Syntax

```
JVMMODULELOCATION pathToJVM
```

Where *pathToJVM* is a fully-qualified path and filename of a Java Virtual Machine library to be used by Essbase.

Description

If you do not include this command in the `essbase.cfg` file, or if you include this command with an incorrect path and filename, Essbase searches the PATH (library path on UNIX systems) for a version of the JVM library and uses the first version that it finds. If you include this command without any parameters, Java Virtual Machine functions, including custom-defined macros and custom-defined functions in the Calculator module, are disabled in the product.

Notes

For more information about setting up the Java Virtual Machine, see the *Oracle Hyperion Enterprise Performance Management System Installation and Configuration Guide*.

Example

```
JVMMODULELOCATION C:\Hyperion\common\JRE\Sun\1.5.0\bin\client\jvm.dll
```

```
// The following statement (with no parameters) disables JVM-dependent functions
```

```
JVMMODULELOCATION
```

The path name cannot include spaces. In `essbase.cfg`, a parameter is not followed by a semicolon. Do not enclose the path parameter in quotation marks.

LOCKTIMEOUT

Limits the amount of time a Spreadsheet Add-in user can hold an exclusive lock.

This setting does not apply to aggregate storage databases.

Syntax

```
LOCKTIMEOUT n
```

Where *n* is a number of seconds. The default value is 3600 seconds (60 minutes).

Description

This setting specifies, in seconds, the maximum amount of time a spreadsheet user can hold an exclusive lock on a block. This `essbase.cfg` setting applies to all applications and databases on the Essbase Server, and is meant to specify a default value for newly created or migrated applications. To override this default value for any specific application, specify a value in Administration Services or MaxL.

Example

```
LOCKTIMEOUT 300
```

commits locked data and releases the exclusive lock after the lock has been held for 300 seconds (five minutes).

LOGINFAILUREMESSAGEDETAILED

Provides detailed error messages on user login failure.

Syntax

```
LOGINFAILUREMESSAGEDETAILED
```

Description

When the LOGINFAILUREMESSAGEDETAILED is specified Essbase provides detailed error messages when a user login fails. The detailed error messages differentiate between, for example, errors caused because the user does not exist, and errors caused by an invalid password.

If you do not specify this configuration setting, Essbase provides the message "Login fails due to invalid login credentials" on user login failure, irrespective of the cause of the login failure.

Notes

Specifying the LOGINFAILUREMESSAGEDETAILED setting has security implications as it potentially provides information to individuals who attempt to login to the system without authorization.

Example

```
LOGINFAILUREMESSAGEDETAILED
```

LOGMESSAGELEVEL

Sets the level of messages written to the application log.

Syntax

```
LOGMESSAGELEVEL INFO | WARNING | ERROR
```

Where INFO, WARNING, and ERROR are priority levels:

- INFO—All three types of messages are written to the application log. This is the default.
- WARNING—Only Warning and Error messages are written to the application log.
- ERROR—Only error messages are written to the application log. No Warning or Info messages are written to the application log.

Notes

- This setting affects only the log messages. To control the messages displayed in the console, use “[DISPLAYMESSAGELEVEL](#)” on page 423. To set the same level for both the console and the log, use both settings.
- For more information about the application log, see the *Oracle Essbase Database Administrator's Guide*.

Example

```
LOGMESSAGELEVEL WARNING
```

sets the log message level to Warning. Only Warning and Error messages are written to the application log.

See Also

[SETMSGLEVEL](#)

“[DISPLAYMESSAGELEVEL](#)” on page 423

LROONSHAREDMBR

Specifies whether shared members have Linked Reporting Objects that are unique from those of their corresponding regular members.

This setting does not apply to aggregate storage databases.

Syntax

```
LROONSHAREDMBR TRUE | FALSE
```

- TRUE—LROs related to regular members are unique, and not shared by shared members. This is the default.
- FALSE—Shared members have the same LROs as corresponding regular members.

Description

A Linked Reporting Object (LRO) is an external file, cell note, or URL that you link to a cell in a database. Users can then retrieve the object from the spreadsheet.

With an LROONSHAREDMBR setting of TRUE, Essbase makes shared member LROs unique from the LROs of regular members.

For example, assume the LROONSHAREDMBR option is FALSE. If you link an LRO to the data cell related to Diet Colas (100-20) under the parent member Colas (100), the corresponding data cell for Diet Colas (100-20) under the parent member Diet shares the same LRO.

Example

```
LROONSHAREDMBR FALSE
```

MAXERRORMBRVERIFYREPORT

Determines the maximum number of members on which Essbase should report errors during outline verification.

Syntax

```
MAXERRORMBRVERIFYREPORT n
```

where *n* the number of members. The default is 500.

Description

MAXERRORMBRVERIFYREPORT limits the number of members upon which Essbase performs error reporting during outline verification. Setting a limit helps avoid performance overhead when a large number of members may cause outline verification errors.

Example

```
MAXERRORMBRVERIFYREPORT 25
```

MAXFORMULACACHESIZE

Applies only to aggregate storage databases. Specifies the maximum size of the aggregate storage 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, specify the application name.
- *dbname*—Optional. To set the cache size maximum for a specific database, specify 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.

Description

If the amount of cache that Essbase sets aside for calculating members of aggregate storage outlines is insufficient, the following error is generated: "ERROR - 1200601 - Not enough memory for formula execution. Set MAXFORMULACACHESIZE configuration parameter to [*n*] and try again." The error recommends a value to use with the MAXFORMULACACHESIZE setting.

Notes

- The entire specified amount is not used unless needed.
- It is recommended that you use this setting only in response to error 1200601.

Example

```
MAXFORMULACACHESIZE 2048
```

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

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
```

Where *n* is 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.

See Also

[“SERVERTHREADS” on page 496](#)

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.

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” on page 469](#) configuration setting

[“ENABLE_DIAG_TRANSPARENT_PARTITION” on page 438](#) 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. 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 1 million (1000000) cells, which requires 8 MB of memory.

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” on page 468](#) configuration setting

[“ENABLE_DIAG_TRANSPARENT_PARTITION” on page 438](#) configuration setting

MDXFORMULARECURSIONLIMIT

When set to false, does not prevent the Essbase Server from going beyond 31 MDX formula execution levels.

Syntax

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

- *appname*—Optional. Specifies the application for which to set or remove the limit. 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 set the limit. If you specify a value for *dbname* but do not specify a value for *appname*, your specification is ignored.
- TRUE—Imposes a limit of 31 on the number of MDX formula execution levels. The default setting is TRUE.
- FALSE—Imposes no limit on the number of MDX formula execution levels.

Description

MDXLIMITFORMULARECURSION limits the number of execution levels of MDX calculated members or formulas. MDX calculated member or formula execution may be recursive (for example, a formula can refer to itself, or a calculated member can refer to itself). By default, Essbase limits the number of MDX formula execution levels, because formulas with excessive execution levels may lead to stack overflow errors and crash the server. However, setting MDXLIMITFORMULARECURSION to FALSE prevents Essbase from imposing the limitation. You can use this setting when you know that a recursive execution in a formula/calculated member will eventually terminate, and you wish to have a recursion depth greater than 31.

If an MDX formula reaches 31 execution levels and MDXLIMITFORMULARECURSION is not set, or is set to TRUE, Essbase stops processing that formula and writes error messages in the application log. If a formula reaches 31 execution levels and MDXLIMITFORMULARECURSION is set to FALSE, Essbase continues processing that formula.

Caution! Before setting MDXLIMITFORMULARECURSION to FALSE, be sure that the MDX formulas in the outline are not infinitely recursive; for example, be sure that formulas do not depend on each other. Infinite formula recursion may crash the server.

MEMSCALINGFACTOR

Enables you to set data cache and data file cache sizes to values greater than 4GB. Applies when running Essbase on 64-bit platforms.

Syntax

MEMSCALINGFACTOR *appname dbname n*

- *appname*—Application name
- *dbname*—Database name
- *n*—The factor used to create data cache and data file cache settings greater than 4 GB. The data cache and data file cache settings in Essbase clients (Administration Services, MaxL, and API) are multiplied by an integer, *n*. The resulting values are the actual cache sizes used by Essbase.

Description

When running Essbase on 64-bit platforms, optimal data cache and data file cache settings may be larger than 4 GB. Although you cannot specify settings larger than 4 GB in Essbase clients, you can enable larger settings using the MEMSCALINGFACTOR configuration setting. When MEMSCALINGFACTOR is enabled, cache settings are multiplied by the factored amount, *n*. The resulting values are the actual cache sizes used by Essbase.

Notes

Both the data cache and the data file cache are scaled using the same factor.

Example

If the data cache setting is 200 MB and, in the `essbase.cfg` file, MEMSCALINGFACTOR is set as follows:

```
MEMSCALINGFACTOR Sample Basic 32
```

The data cache size for Sample Basic is 200 MB x 32 = 6400 MB.

See Also

[“DATACACHESIZE” on page 412](#)

[“DATAFILECACHESIZE” on page 415](#)

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” on page 396](#)

[“PARCALCMULTIPLEBITMAPMEMOPT” on page 479](#)

NETBINDRETRYDELAY

Specifies the amount of time, in milliseconds, that the application server retries on a bind failure.

Syntax

```
NETBINDRETRYDELAY n
```

Where *n* is an integer value, expressed in milliseconds. The default value is 10 seconds. The minimum value is 0.

Example

```
NETBINDRETRYDELAY 5
```

Causes the application server network to retry on a bind failure after 5 milliseconds.

See Also

[“NETDELAY” on page 473](#)

[“NETRETRYCOUNT” on page 474](#)

[“NETTCPCONNECTRETRYCOUNT” on page 475](#)

NETDELAY

Specifies the network request delay time.

Syntax

```
NETDELAY n
```

Where *n* is an integer value of 100 or above, expressed in milliseconds. The default value is 200 milliseconds.

Description

This setting defines the network request delay time in milliseconds. This is the amount of time an unsuccessful operation waits before Essbase retries the operation.

Example

```
NETDELAY 500
```

See Also

[“NETBINDRETRYDELAY” on page 473](#)

[“NETRETRYCOUNT” on page 474](#)

[“NETTCPCONNECTRETRYCOUNT” on page 475](#)

NETRETRYCOUNT

Specifies the number of attempts Essbase is allowed to make a network connection before failing and reporting an error.

Syntax

```
NETRETRYCOUNT n
```

Where *n* is an integer value. The default value is 600 retries. The minimum value is 300.

Example

```
NETRETRYCOUNT 400
```

See Also

[“NETBINDRETRYDELAY” on page 473](#)

[“NETDELAY” on page 473](#)

[“NETTCPCONNECTRETRYCOUNT” on page 475](#)

NETSSLHANDSHAKETIMEOUT

Specifies the maximum time that Essbase Client should wait for Essbase Agent to respond to a secure session request before timing out.

Syntax

```
NETSSLHANDSHAKETIMEOUT n
```

Where *n* is the number of milliseconds expressed as a positive integer. The default is 10000 milliseconds (10 seconds).

Description

Use this setting to specify the maximum number of milliseconds that Essbase clients should wait for a response to a secure session request before timing out.

Notes

- The minimum value is 100 milliseconds; values less than the minimum are ignored.
- The SSL handshake may timeout due to network congestion, or because the connection modes at either end are mismatched (for example, a client in Clear mode tries to connect to the secure port of Essbase Agent by mistake).

Example

```
NETSSLHANDSHAKETIMEOUT 20000
```

The SSL handshake fails after 20,000 milliseconds if Essbase Agent does not respond to the secure session request.

See Also

[“AGENTSECUREPORT” on page 388](#)

[“CLIENTPREFERREDMODE” on page 411](#)

[“ENABLECLEARMODE” on page 439](#)

[“ENABLESECUREMODE” on page 440](#)

[“SSLCIPHERSUITES” on page 502](#)

[“WALLETPATH” on page 515](#)

For information on implementing SSL, see the *Oracle Hyperion Enterprise Performance Management System Security Administration Guide*.

NETTCPCONNECTRETRYCOUNT

Specifies the number of attempts a client will make to connect to a TCP/IP network before failing and reporting an error.

Syntax

```
NETTCPCONNECTRETRYCOUNT n
```

Where *n* is an integer value. The default value is 3.

Notes

Some causes of connection failures are, for example, network congestion, server inaccessibility, and network interruption.

Example

```
NETTCPCONNECTRETRYCOUNT 100
```

See Also

[“NETRETRYCOUNT” on page 474](#)

[“NETDELAY” on page 473](#)

[“NETBINDRETRYDELAY” on page 473](#)

NOMSGLOGGINGONDATAERRORLIMIT

Prevents data load or dimension build errors from being written to the application log after the limit described by the value of DATAERRORLIMIT is reached.

Syntax

```
NOMSGLOGGINGONDATAERRORLIMIT TRUE | FALSE
```

The default value is FALSE.

Description

This setting controls the maximum number of error messages written to the data load error log per data load and the dimension build error log per dimension build. This configuration setting, NOMSGLOGGINGONDATAERRORLIMIT, stops any data load or dimension build error messages from being written to the application log after the DATAERRORLIMIT value has been reached.

The default value for DATAERRORLIMIT is 1000, so if you do not set DATAERRORLIMIT, only the first 1000 errors will be written to the data load error log or the dimension build error log.

Example

```
DATAERRORLIMIT 50000  
NOMSGLOGGINGONDATAERRORLIMIT TRUE
```

Sets the limit on data load or dimension build error messages written to the error log at 50,000, and further prevents any error messages after the first 50,000 from being written to the application log.

See Also

[“DATAERRORLIMIT” on page 413](#)

NUMBEROFSECFILEBACKUPS

Specifies the maximum number of security backup files (*essbase_timestamp.bak*) that Essbase creates and maintains. When the limit is exceeded, Essbase deletes the security backup file with the oldest timestamp and creates the latest backup file.

Syntax

```
NUMBEROFSECFILEBACKUPS n
```

n—Specifies an integer between 2 and 10.

The default value is 2.

Note: You can configure the interval in which Essbase creates backup security files, and whether Essbase automatically loads a valid backup security file at startup, if the `essbase.sec` file is invalid.

Example

```
NUMBEROFSECFILEBACKUPS 5
```

See Also

[“ENABLESWITCHTOBACKUPFILE” on page 440](#)

[“SECFILEBACKUPINTERVAL” on page 490](#)

NUMERICPRECISION

Sets the number of precision digits used by Report Writer for numerical comparison.

Syntax

```
NUMERICPRECISION n
```

Where *n* is the number of precision digits to be considered in the numerical comparison. Acceptable values for *n* are -1 through 15. A value of -1 indicates a full comparison. The default value is 4.

Description

This setting defines the number of precision digits used by Report Writer for numerical comparison.

The numeric comparison function subtracts one value from the other, and compares the absolute value of the result with 10^{-n} . If 10^{-n} is greater than the absolute value of the subtraction result, the numbers are equal.

Notes

- A value of -1 indicates a full comparison.
- For information about Report Writer, see the *Oracle Essbase Database Administrator's Guide*.

Example

Suppose we compare the values 3.289999 and 3.290000 with a numeric precision of 2:

```
NUMERICPRECISION 2
```

Is $3.289999 == 3.290000$ given a numeric precision of 2?

$|3.289999 - 3.290000| = 0.000001$ (the absolute value)

$10^{-2} = 0.01$

0.01 > 0.000001, so the numbers are equal.

See Also

[RESTRICT Report Writer Command](#)

OUTLINECHANGELOG

Controls whether Essbase keeps a history of outline modifications.

Syntax

```
OUTLINECHANGELOG TRUE | FALSE
```

- TRUE—Essbase logs outline changes into the file *database_name.olg*.
- FALSE—Essbase does not log outline changes. The default is FALSE.

Description

If OUTLINECHANGELOG is set to TRUE, Essbase logs all outline changes into the file *database_name.olg*. Database administrators can review the outline revision history in the *.olg* file and gather enough information to roll back changes if needed.

Each database contains a separate outline change log file in the same location as the database. The file is stored in the database directory of the Essbase Server installation.

The data format of the outline change log is:

- Date and time of outline modification
- Name of the user who made the change
- Type of change the user made
- Details describing the type of change made

Notes

- During a restructure, Essbase holds outline change information in memory until all updates have been made to the outline change log. Turning on the outline change log might affect your restructure performance, particularly after dimension builds of several hundred or more members.
- To set the size of the outline change log, use the [“OUTLINECHANGELOGFILESIZE” on page 479](#) parameter in your *essbase.cfg* file.

Example

```
OUTLINECHANGELOG TRUE
```

See Also

[“OUTLINECHANGELOGFILESIZE” on page 479](#)

[“SILENTOTLQUERY” on page 497](#)

OUTLINECHANGELOGFILESIZE

Sets the maximum file size of the outline change log.

Syntax

```
OUTLINECHANGELOGFILESIZE n
```

Where *n* is the number of bytes to allocate for the change log. The default is 64,000 bytes. The minimum is 8,092 bytes. The maximum is 2 megabytes.

Description

This setting sets the maximum file size of the outline change log in bytes. When the outline change log reaches the maximum file size, Essbase copies the contents of the file to a separate backup file with the same name as the outline change log file (`database_name.olg`), but with an `.olb` extension.

Notes

- The outline change log is disabled by default. To enable it, use the [“OUTLINECHANGELOG” on page 478](#) parameter in your `essbase.cfg` file.
- The outline change log file is located in the database directory of the Essbase Server installation. It is named in the format `database_name.olg`.
- The default, minimum, and maximum file sizes for the backup file are the same as the file sizes specified for the outline change log file.
- Each time the outline change log file reaches its maximum file size, Essbase clears the outline change log and replaces the backup file with a backup of the current outline change log.

Example

```
OUTLINECHANGELOGFILESIZE 8092
```

See Also

[“OUTLINECHANGELOG” on page 478](#)

[“SILENTOTLQUERY” on page 497](#)

PARCALMULTIPLEBITMAPMEMOPT

Optimizes memory use when using multiple bitmap mode during parallel calculation.

This setting does not apply to aggregate storage databases.

Syntax

```
PARCALMULTIPLEBITMAPMEMOPT TRUE | FALSE
```

- TRUE—Memory usage is optimized.
- FALSE—Memory usage is not optimized.

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” on page 472](#).

Example

```
PARCALCMULTIPLEBITMAPMEMOPT TRUE
```

See Also

[“CALCCACHE” on page 396](#)

[“MULTIPLEBITMAPMEMCHECK” on page 472](#)

PERSISTUSERATLOGIN

When a user logs on to Essbase, specifies whether to add the user to the `essbase.sec` security file, if the user does not already exist in the file.

Syntax

```
PERSISTUSERATLOGIN TRUE | FALSE
```

- TRUE—Essbase adds the user to the security file, and tracks user information (such as the time the user last logged into Essbase) and named connections.
- FALSE—The user is not added to the security file.

Example

```
PERSISTUSERATLOGIN TRUE
```

PIPEBUFFERSIZE

Sets the size of the buffer used for communication between the Spreadsheet Add-in extractor and Report Writer.

Syntax

```
PIPEBUFFERSIZE n
```

Where *n* is an integer value from 2,048 to 65,534, expressed in bytes. The default value is 4K (4,096 bytes).

Description

This setting determines the size of the buffer used for communication between the Spreadsheet Add-in extractor and Report Writer on the network.

Example

```
PIPEBUFFERSIZE 20000
```

defines a 20-kilobyte buffer to store pipes.

PORTINC

Specifies the value of the increment in between port numbers used by the Essbase agent process.

Syntax

```
PORTINC n
```

Where *n* is the value of *n* specifies the increment between port numbers that the Agent used to try and find an available port. The default value is 1.

Description

This setting specifies the increment value between ports used by the Agent when it tries to find an available port.

You may wish to change the default for many reasons. These are two common reasons:

- The default port, 33768, is inappropriate for your site.
- You may wish to install a second Agent on a single computer to facilitate testing. see SERVERPORTEND and the related configuration settings to assign the second Agent to a different port than the first. Use SERVERPORTEND along with AGENTPORT, SERVERPORTBEGIN, and PORTINC.

Caution! More than one Agent per computer should not be used in production systems.

Notes

- You must insert these settings in both the configuration file for the Essbase Server computer and the configuration file for the client computer.
- You must perform several other steps in order to enable multiple agents on one computer. See the *Oracle Essbase Database Administrator's Guide* for instructions.

Example

```
AGENTPORT 1478  
SERVERPORTBEGIN 32470  
SERVERPORTEND 32600  
PORTINC 5
```

This example would produce these results:

- AGENTPORT sets the port that the additional Agent will use at 1478.
- SERVERPORTBEGIN sets the value that the first server process will try to use for a port at 32470.

- SERVERPORTEND sets the highest port number value this installation can use.
- PORTINC controls the increment value used for each port. In this example, if the first server process was able to use port number 32470, then the next process would use 32475.

See Also

[“AGENTPORT” on page 387](#)

[“SERVERPORTBEGIN” on page 493](#)

[“SERVERPORTEND” on page 494](#)

[“PORTUSAGELOGINTERVAL” on page 482](#)

PORTUSAGELOGINTERVAL

Enables Essbase Server to log, at a specified interval, the number of ports being used.

Syntax

```
PORTUSAGELOGINTERVAL n
```

Where *n* represents the number of minutes between each check of the number of ports in use. The value of *n* can be any whole number from 1 - 60, with five as the recommended minimum and default value. Essbase ignores any portion of a non-whole number. For example, 2.5 is evaluated as 2 minutes. Statistics are written to the log immediately after each check.

Description

PORTUSAGELOGINTERVAL enables you to set an interval at which to log the number of ports being used. By analyzing the information in the log, you can monitor port utilization and identify a need for more ports before end users are unable to connect.

To enable Essbase Server to check port use statistics and write those statistics to the log:

1. Edit the server configuration file `essbase.cfg` to include the PORTUSAGELOGINTERVAL setting.
2. Restart Essbase Server.
3. View the Essbase Server Log file. You will see entries similar to the following output:

```
[Mon Apr 22 00:48:50 2003]Local/ESSBASE0///Info(1056214)
[3] ports in use, [10] ports allowed
```

Examples

```
PORTUSAGELOGINTERVAL 10
```

Essbase writes the port use statistics to the Essbase Server log every 10 minutes.

```
PORTUSAGELOGINTERVAL
```

Essbase writes the port use statistics to the Essbase Server log every five minutes (the default value).

Essbase ignores the non-whole portion of the number and writes the port use statistics to the Essbase Server log every six minutes.

See Also

[“SERVERPORTBEGIN” on page 493](#)

[“SERVERPORTEND” on page 494](#)

[“PORTINC” on page 481](#)

[“AGENTPORT” on page 387](#)

PRELOADALIASNAMESPACE

Applies only to aggregate storage databases. Determines whether the namespace for alias names is preloaded at database startup.

Syntax

```
PRELOADALIASNAMESPACE TRUE | FALSE
```

- TRUE—The default. The namespace for alias names is preloaded at database startup.
- FALSE—The namespace for alias names is paged into memory as needed.

Description

PRELOADALIASNAMESPACE determines whether the namespace for alias names is preloaded at database startup. Preloading the namespace may improve performance but uses additional memory. The alias namespace is used to search for an alias by name. The search occurs during data load, during report and spreadsheet queries, and during MDX queries.

Example

```
PRELOADALIASNAMESPACE FALSE
```

See Also

[“PRELOADMEMBERNAMESPACE” on page 483](#)

PRELOADMEMBERNAMESPACE

Applies only to aggregate storage databases. Determines whether the namespace for member names is preloaded at database startup.

Syntax

```
PRELOADMEMBERNAMESPACE TRUE | FALSE
```

- TRUE—The default. The namespace for member names is preloaded at database startup.

- FALSE—The namespace for member names is paged into memory as needed.

Description

PRELOADMEMBERNAMESPACE determines whether the namespace for member names is preloaded at database startup. Preloading the namespace may improve performance but uses additional memory. The member namespace is used to search for a member by name. The search occurs during data load, during report and spreadsheet queries, and during MDX queries.

Example

```
PRELOADMEMBERNAMESPACE FALSE
```

See Also

[“PRELOADALIASNAMESPACE” on page 483](#)

PRELOADUDANAMESPACE

Determines whether the namespace for UDAs is preloaded at application startup.

Syntax

```
PRELOADUDANAMESPACE appname TRUE | FALSE
```

- *appname*—Application for which the UDA namespace is preloaded at start up.
- TRUE—Namespace for UDAs is preloaded at application startup.
- FALSE—The default. Namespace for UDAs is paged into memory, as needed.

Description

Because querying member sets by UDA can take a long time in large outlines (for example, with one million or more members), in which many members (for example, half a million members) are assigned to one UDA, and login time can be slow for users with filters containing UDAs, preloading the UDA namespace may improve performance. However, preloading the UDA namespace uses additional memory. To calculate the additional memory consumption, use these formulas:

32-bit platforms:

Four additional bytes are used per UDA associated with a member, plus 12 bytes per distinct UDA. The formula:

$$(\#_of_members \times \#_of_UDAs_per_member \times 4 \text{ bytes}) + (\#_of_distinct_UDAs \times 12 \text{ bytes})$$

64-bit platforms:

Eight additional bytes are used per UDA associated with a member, plus 24 bytes per distinct UDA. The formula:

$$(\#_of_members \times \#_of_UDAs_per_member \times 8 \text{ bytes}) + (\#_of_distinct_UDAs \times 24 \text{ bytes})$$

For example, for an outline with 1,000,000 members and 500,000 distinct UDAs, in which two UDAs are associated with each member, the additional memory needed is:

32-bit platforms:

(1,000,000 members x 2 UDAs associated to each member x 4 bytes) + (500,000 distinct UDAs x 12 bytes) 14 MB

64-bit platforms:

(1,000,000 members x 2 UDAs associated to each member x 8 bytes) + (500,000 distinct UDAs x 24 bytes) 28 MB

Example

```
PRELOADUDANAMESPACE ASOsamp TRUE
```

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 attributes 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 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.
- If a query involves one or more Hybrid Analysis Relational Sources, QRYGOVEXECBLK is disabled upon encountering the first relational member.

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 that correspond to the `essbase.cfg` file containing this setting.

See Also

[“QRYGOVEXECTIME” on page 486](#)

[“HAMAXQUERYROWS” on page 451](#)

[“HAMAXQUERYTIME” on page 452](#)

For more information about the application log, see the *Oracle Essbase Database Administrator's Guide*.

QRYGOVEXECTIME

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

Syntax

QRYGOVEXEETIME [*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 QRYGOVEXEETIME. 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 QRYGOVEXEETIME. 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. You must specify this parameter or Essbase Server ignores QRYGOVEXEETIME. If do not specify *appname* or *dbname*, the query time limit applies to the entire server.

Description

QRYGOVEXEETIME 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 QRYGOVEXEETIME to prevent these types of queries:

- A long-running query against a database that accesses attributes 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 QRYGOVEXEETIME, 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 QRYGOVEXEETIME.
- 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.

- If a query involves one or more Hybrid Analysis Relational Sources, QRYGOVEXEETIME is disabled upon encountering the first relational member.

Example

```
QRYGOVEXEETIME 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.

```
QRYGOVEXEETIME 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 that correspond to the `essbase.cfg` file containing this setting.

See Also

[“QRYGOVEXEETIME” on page 485](#)

[“HAMAXQUERYROWS” on page 451](#)

[“HAMAXQUERYTIME” on page 452](#)

For more information about the application log, see the *Oracle Essbase Database Administrator's Guide*.

REPLAYSECURITYOPTION

Specifies the user security settings that are used when replaying logged transactions.

Syntax

```
REPLAYSECURITYOPTION n
```

n—An integer that specifies the user security setting. Valid values are as follows:

- 1—(Default) Specifies the security settings of the user who originally performed the transaction. If that user no longer exists or that user's username was changed, the replay operation will fail.

Oracle does not recommend renaming another user with the name of the original user, as the security settings of the renamed user might not match those of the original user and the transaction might be played with the incorrect security settings.

- 2—Specifies the security settings of the administrator performing the replay operation.
- 3—Specifies the security settings of the user who originally performed the transaction. If that user no longer exists or that user's username was changed, the security settings of the administrator performing the replay operation are used.

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

See Also

[Alter Database MaxL statement](#)

[“TRANSACTIONLOGLOCATION” on page 510](#) configuration setting

[“TRANSACTIONLOGDATALOADARCHIVE” on page 508](#) configuration setting

REPLICATIONASSUMEIDENTICALOUTLINE

Optimizes the replication of a partitioned, aggregate storage database when the aggregate storage database is the target and a block storage database is the source, and the two outlines are identical.

The setting affects only the target aggregate storage application (not the source block storage application) and does not apply to block storage replication.

REPLICATIONASSUMEIDENTICALOUTLINE can be enabled at the server, application, or database level. You can also use the **alter database** MaxL statement with the **replication_assume_identical_outline** grammar to enable replication optimization at the database level only.

Syntax

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

- *appname*—Optional. Specifies the application to be enabled for replication optimization.
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*, to be enabled for replication optimization.
If you specify a value for *dbname* but do not specify a value for *appname*, your specification is ignored, and replication optimization is enabled for all applications and databases on Essbase Server.
- TRUE | FALSE—Specifies whether to enable or disable replication optimization.

You must restart Essbase Server to initialize changes to the configuration file.

Example

```
REPLICATIONASSUMEIDENTICALOUTLINE AsoSamp.Sample TRUE
```

Optimizes the replication of the ASOsamp.Sample database, when it is the target of a replicated partition and its outline is identical to the outline of the source block storage database.

See Also

[alter database \(aggregate storage\) MaxL statement](#)

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

SECFILEBACKUPINTERVAL

Specifies the maximum amount of time (in seconds) that Essbase waits before creating a backup of the `essbase.sec` file. Named, `essbase_timestamp.bak`, Essbase can create and maintain from 2 to 10 backup security files.

Syntax

```
SECFILEBACKUPINTERVAL x
```

n—Specifies the amount of time in seconds.

The default value is 300 seconds (which is five minutes). A value of 0 means that Essbase won't perform this check.

Note: You can configure the number of backup security files that Essbase creates and maintains, and whether Essbase automatically loads a valid backup security file at startup, if the `essbase.sec` file is invalid.

Example

```
SECFILEBACKUPINTERVAL 600
```

See Also

[“NUMBEROFSECFILEBACKUPS” on page 476](#)

[“ENABLESWITCHTOBACKUPFILE” on page 440](#)

SECURITYFILECOMPACTIONPERCENT

Specifies the percentage of obsolete space in the security file (`essbase.cfg`) that is a factor in triggering compaction of that file.

Syntax

```
SECURITYFILECOMPACTIONPERCENT n
```

Where *n* is the percentage limit of obsolete space that will trigger compaction of the security file.

n is an integer between 10 and 100. The recommended value is 30.

Description

Changing or deleting the following Essbase Server security entities can cause fragmentation in the security file (`essbase.sec`): filters, users, groups, applications, databases, substitution variables, disk volumes, passwords, and other Essbase Server objects.

Essbase compacts the security file automatically each time the Agent is stopped. You can use the SECURITYFILECOMPACTIONPERCENT configuration setting to trigger compaction of the security file when the agent is still running and no Agent activity has occurred for the period of time specified by the "timeout" Essbase Server property.

Notes

- The timeout period is a server property defined, per user, in Administration Services or MaxL. Compaction based on the SECURITYFILECOMPACTIONPERCENT configuration setting occurs only when the timeout has caused all users to be logged out.
- Once compaction is initiated through this configuration setting, if you log back in and perform a task that requires Agent activity, the task will be delayed until compaction is completed.
- You can force compaction using the COMPACT Agent command or the `alter system MaxL` statement.
- See the appropriate documentation for details.

Example

```
SECURITYFILECOMPACTIONPERCENT 30
```

See Also

[display system security file fragmentation_percent](#); (MaxL)

[alter system compact security file](#) (MaxL)

COMPACT (Agent command), in the *Oracle Essbase Database Administrator's Guide*

SERVERLEASEEXPIRATIONTIME

Sets the maximum amount of time that Essbase Server can own a lease before the lease is terminated.

Syntax

```
SERVERLEASEEXPIRATIONTIME n
```

Where *n* is an integer specifying the number of seconds before a lease expires. The default value is 20.

Example

```
SERVERLEASEEXPIRATIONTIME 20
```

See Also

[“AGENTLEASEEXPIRATIONTIME”](#) on page 384

[“AGENTLEASEMAXRETRYCOUNT”](#) on page 385

[“AGENTLEASERENEWALTIME”](#) on page 385

[“SERVERLEASEEXPIRATIONTIME”](#) on page 492

[“SERVERLEASEMAXRETRYCOUNT”](#) on page 492

[“SERVERLEASERENEWALTIME”](#) on page 493

SERVERLEASEMAXRETRYCOUNT

Specifies the number of times that Essbase Server attempts to acquire or renew a lease. If the attempts are unsuccessful, the server terminates itself.

Syntax

```
SERVERLEASEMAXRETRYCOUNT n
```

Where *n* is an integer. The default value is 5.

Example

```
SERVERLEASEMAXRETRYCOUNT 5
```

See Also

[“AGENTLEASEEXPIRATIONTIME” on page 384](#)

[“AGENTLEASEMAXRETRYCOUNT” on page 385](#)

[“AGENTLEASERENEWALTIME” on page 385](#)

[“SERVERLEASEEXPIRATIONTIME” on page 492](#)

[“SERVERLEASEMAXRETRYCOUNT” on page 492](#)

[“SERVERLEASERENEWALTIME” on page 493](#)

SERVERLEASERENEWALTIME

Specifies the time interval after which Essbase Server renews its lease.

Syntax

```
SERVERLEASERENEWALTIME n
```

Where *n* is an integer specifying the number of seconds available to reestablish ownership after a lease expires. The default value is 10.

Example

```
SERVERLEASERENEWALTIME 10
```

See Also

[“AGENTLEASEEXPIRATIONTIME” on page 384](#)

[“AGENTLEASEMAXRETRYCOUNT” on page 385](#)

[“AGENTLEASERENEWALTIME” on page 385](#)

[“SERVERLEASEEXPIRATIONTIME” on page 492](#)

[“SERVERLEASEMAXRETRYCOUNT” on page 492](#)

SERVERPORTBEGIN

Specifies the first port number that Essbase tries to use for its first application process (ESSVR).

Syntax

```
SERVERPORTBEGIN n
```

Where *n* specifies the port number that Essbase tries to use for its first application process. This port number should not be in use by any other process. The default value is 32768.

Description

SERVERPORTBEGIN specifies the first port that Essbase tries to use for the first application process it tries to start.

You may want to change the default for many reasons. These are two common reasons:

- The default port, 1423, is inappropriate for your site.
- You intend to install a second Agent on a single computer to facilitate testing. Use `SERVERPORTBEGIN` and the related configuration settings to assign the second Agent to a different port than the first. Use `SERVERPORTBEGIN` with `AGENTPORT`, `SERVERPORTEND`, and `PORTINC`.

Caution! More than one Agent per computer should not be used in production systems.

Notes

- You must perform several other steps in order to enable multiple agents on one computer. See the *Oracle Essbase Database Administrator's Guide* for instructions.
- `SERVERPORTBEGIN` and `SERVERPORTEND` cannot have the same value.

Example

```
AGENTPORT 1478
SERVERPORTBEGIN 32470
SERVERPORTEND 32600
PORTINC 5
```

This example would produce these results:

- `AGENTPORT` sets the port that the additional Agent will use at 1478.
- `SERVERPORTBEGIN` sets the value that the first application process will try to use for a port at 32470.
- `SERVERPORTEND` sets the highest port number value this installation can use.
- `PORTINC` controls the increment value used for each port. In this example, if the first application process was able to use port number 32470, then the next process would use 32475.

See Also

[“AGENTPORT” on page 387](#)

[“SERVERPORTEND” on page 494](#)

[“PORTINC” on page 481](#)

[“PORTUSAGELOGINTERVAL” on page 482](#)

SERVERPORTEND

Specifies the highest value that Essbase tries to use for a port when it starts an application process (ESSVR). If the value is unavailable, the application process fails.

Syntax

```
SERVERPORTEND n
```

Where n specifies the highest value for a port number that Essbase tries to use for a application process. If the port is unavailable, the application process fails. This port number should not be in use by any other process. The default value is 33768.

Description

SERVERPORTEND specifies the highest port number that Essbase uses when trying to start an application process.

You may want to change the default for many reasons. These are two common reasons:

- The default port, 33768, is inappropriate for your site.
- You want to install a second Agent on a single computer to facilitate testing. Use SERVERPORTEND and the related configuration settings to assign the second Agent to a different port than the first. Use SERVERPORTEND along with AGENTPORT, SERVERPORTBEGIN, and PORTINC.

Caution! More than one Agent per computer should not be used in production systems.

Notes

- You must perform several other steps in order to enable multiple agents on one computer. See the *Oracle Essbase Database Administrator's Guide* for instructions.
- SERVERPORTBEGIN and SERVERPORTEND cannot have the same value.

Example

```
AGENTPORT 1478
SERVERPORTBEGIN 32470
SERVERPORTEND 32600
PORTINC 5
```

This example would produce these results:

- AGENTPORT sets the port that the additional Agent will use at 1478.
- SERVERPORTBEGIN sets the value that the first application process will try to use for a port at 32470.
- SERVERPORTEND sets the highest port number value this installation can use.
- PORTINC controls the increment value used for each port. In this example, if the first server process was able to use port number 32470, then the next process would use 32475.

See Also

[“AGENTPORT” on page 387](#)

[“SERVERPORTBEGIN” on page 493](#)

[“PORTINC” on page 481](#)

[“PORTUSAGELOGINTERVAL” on page 482](#)

SERVERTHREADS

Overrides the default value for the number of threads that applications may spawn.

Syntax

`SERVERTHREADS [application_name] n`

- *application_name*—Optional. If you specify an application, all the databases in that application are affected by the `SERVERTHREADS` setting. If you leave out the application name parameter, the setting applies to the entire Essbase Server.
- *n*—The number of threads the application may produce:
 - Between 20 and 500 on 32-bit platforms.
 - Between 20 and 1024 on 64-bit platforms.

Specify an integer between 20 and the maximum, inclusive. If you use the `.CFG` setting to specify a number lower than 20, it is interpreted as 20. If you use the `.CFG` setting to specify a number higher than the maximum, it is interpreted as 500 (on 32-bit platforms), or 1024 (on 64-bit platforms).

See Notes below.

Description

`SERVERTHREADS` overrides the default value for the number of threads that application processes may spawn. The default value is defined by the number of licensed ports. Use this setting to make the maximum number of threads higher than the default value. See Notes.

When a transaction is requested, the application assigns a thread to the transaction and releases the thread when the transaction is complete.

Notes

- The default value of `SERVERTHREADS` depends on the number of available ports on the Essbase Server. Note that one additional port is available for the system administrator, so that for 5 licensed ports, 6 available ports are shown when you type `PORTS` in the Agent window.
 - 5 or fewer ports—5 default threads
 - 6 to 11 ports—10 default threads
 - 11 or more ports—20 default threads
- Although the actual maximum value you can set is 500 (or 1024 on 64-bit platforms), the maximum number of threads an operating system can handle might be much lower. It is strongly recommended that you use the default value. If you want to set this parameter to a value higher than the default, check with your system administrator, as higher values can significantly consume system resources.
- If your Essbase Server computer freezes while running multiple reports at the same time, increase the value of `SERVERTHREADS` by one for each report you run.

Example

```
SERVERTHREADS 25
```

Allows all applications to spawn up to 25 threads.

```
SERVERTHREADS Sample 100
```

Allows the Sample application to spawn up to 100 threads.

See Also

[“AGENTDELAY” on page 382](#)

[“AGENTTHREADS” on page 388](#)

SILENTOTLQUERY

Controls whether Essbase keeps a history of outline queries in the application log file.

Syntax

```
SILENTOTLQUERY [appName [dbName]] TRUE | FALSE
```

- *appName*—Optional. If you supply an application name, the TRUE or FALSE setting applies to all databases within the named application.
- *dbName*—Optional. If you supply a database name and an application name, the TRUE or FALSE setting applies only to the named database. If you supply a database name, you must also supply an application name.
- TRUE—Essbase does not log outline queries in the application log file.
- FALSE—Essbase logs outline queries in the application log file. The default is FALSE.

Example

```
SILENTOTLQUERY TRUE
```

See Also

[“OUTLINECHANGELOG” on page 478](#)

[“OUTLINECHANGELOGFILESIZE” on page 479](#)

SPLITARCHIVEFILE

When backing up a database to an archive file, specifies whether to split the archive file into multiple files (with each file being no larger than 2 GB) or to create a single, large archive file (the size of which is limited only by disk space).

Syntax

```
SPLITARCHIVEFILE TRUE | FALSE
```

The default value is FALSE.

- TRUE—Creates multiple database archive files.
- FALSE—(Default)Creates a single database archive file.

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

Description

Splitting the archive file into smaller, multiple files is useful if you cannot use large files or the file-transfer tools that you use cannot handle large files.

The first (or main) archive file that Essbase creates uses the filename that you specify (for example, `samplebasic.arc`). When the main archive file reaches the 2 GB limit, Essbase creates another archive file. In naming the other archive files, Essbase increments the main archive filename with “_x”, where x is an integer (starting with 1). Using the `samplebasic.arc` example, if three archive files are created when backing up the Sample.Basic database, the filenames would be:

```
samplebasic.arc  
samplebasic_1.arc  
samplebasic_2.arc
```

All archive files are created in the directory that you specified when specifying the filename and location of the main archive file.

If you use the default, single-file configuration, Oracle recommends saving archive files to a file system that supports large files. For Windows, the file system must be formatted as NTFS. For UNIX, large file support must be enabled (for example, use the `ULIMIT` setting to specify a specific file size based on the size of the database or set `ULIMIT` to unlimited). See your operating system documentation.

Note: When restoring a database in which the archive file is split into multiple files, Essbase looks for multiple archive files, even if, after the backup, you subsequently set `SPLITARCHIVEFILE` to FALSE for that database. Also, Essbase expects all of a database's archive files (main and split) to be in the same directory.

See Also

[Alter Database](#) MaxL statement

[Query Database](#) MaxL statement

Oracle Hyperion Enterprise Performance Management System Backup and Recovery Guide

SQLFETCHERRORPOPUP

Controls whether an Essbase error is generated when fetching data from a SQL database during a data load or a dimension build. The error will provide a pop-up error message in Administration Services, and will enable error handling using `IFERROR` in MaxL Shell or `ESSCMD`.

Syntax

```
SQLFETCHERRORPOPUP TRUE | FALSE
```

- TRUE—SQL imports generate error messages.
- FALSE—Default value. SQL imports do not generate error messages.

Example

```
SQLFETCHERRORPOPUP TRUE  
SQLFETCHERRORPOPUP FALSE
```

SSAUDIT

Enables spreadsheet update logging, appending to existing logs after archiving.

This setting does not apply to aggregate storage databases.

Syntax

```
SSAUDIT appname [ dbname [ log_path ] ]
```

- *appname*—Application name.
- *dbname*—Optional. Database name.
- *log_path*—Optional. Full directory path where you want the information stored. Do not provide a *log_path* value unless you have also provided a value for *dbname*.

Default behavior:

- If SSAUDIT (or SSAUDITR) is not specified, spreadsheet update logging is not enabled.
- If SSAUDIT (or SSAUDITR) is issued with no arguments, Essbase activates spreadsheet logging for all databases in all applications on the Essbase Server, and puts the log in the default directory: *ARBORPATH\app\appname dbname*.

Use the value `xxxxxx` to indicate "all" for any argument.

You can issue up to ten (total) SSAUDIT and/or “SSAUDITR” on page 500 statements per application.

Description

SSAUDIT enables Essbase to log successfully completed spreadsheet update transactions. The resulting logs can be used as a source of input data upon recovery after archive operations or other server interruptions.

Notes

- SSAUDIT is not available when using Free-Form reporting in Spreadsheet Add-in.
- If you have duplicate database names in different applications, do not store their error logs in the same directory. If you do, the log for one database will be replaced by the log for any subsequent database with the same name.

- SSAUDIT creates two logs for each database:
 - *dbname.atx*, which stores the update transaction records that can be used as the input source for data load
 - *dbname.alg*, which stores history records from every update transaction, including user name, time stamp, and number of updated rows
- Essbase ensures that if you enable spreadsheet logging, updates do not take place without getting logged. If Essbase cannot write to the update logs for any reason, Essbase fails the update transaction and issues an error message.
- SSAUDIT may slow Lock and Send operations.

Example

```
SSAudit xxxxx xxxxx c:\sslog
```

enables logging for all applications and databases, storing the log in the path `c:\sslog`. This example assumes that you do not have duplicate database names (see Notes).

See Also

[“SSAUDITR” on page 500](#), which clears the log after archive.

[alter database begin | end archive \(MaxL\)](#)

[BEGINARCHIVE \(ESSCMD\)](#)

[ENDARCHIVE \(ESSCMD\)](#)

SSAUDITR

Enables spreadsheet update logging, clearing the logs at the end of the archiving process.

Syntax

```
SSAUDITR appname [ dbname [ log_path ] ]
```

- *appname*—Application name.
- *dbname*—Optional. Database name.
- *log_path*— Optional. Full directory path where you want the information stored. Do not provide a *log_path* value unless you have also provided a value for *dbname*.

Default behavior:

- If SSAUDITR (or SSAUDIT) is not specified, spreadsheet update logging is not enabled.
- If SSAUDITR (or SSAUDIT) is issued with no arguments, Essbase activates spreadsheet logging for all databases in all applications on the Essbase Server, and puts the log in the default directory: `ARBORPATH\app\appname dbname`.

Use the value `xxxxxx` to indicate "all" for any argument.

You can issue up to ten (total) SSAUDITR and/or “SSAUDIT” on page 499 statements per application.

Description

SSAUDITR enables Essbase to log successfully completed spreadsheet update transactions. The resulting logs can be used as a source of input data upon recovery after archive operations or other server interruptions.

Notes

- SSAUDITR creates two logs for each database:
 - *dbname.atx*, which stores the update transaction records that can be used as the input source for data load
 - *dbname.alg*, which stores history records from every update transaction, including user name, time stamp, and number of updated rows
- Essbase ensures that if you enable spreadsheet logging, updates do not take place without getting logged. If Essbase cannot write to the update logs for any reason, the update transaction fails and an error message is issued.
- SSAUDITR may slow Lock and Send operations.
- The spreadsheet log file will not be cleared if the database is shut down during archive mode. The database is expected to remain running while in archive mode.

Example

```
SSAuditR demo
```

Enables logging with refresh (clear) for all databases belonging to the Demo application. The log is stored in the default directory.

See Also

“SSAUDIT” on page 499, which does not clear the logs after archive.

[alter database begin | end archive \(MaxL\)](#)

[BEGINARCHIVE \(ESSCMD\)](#)

[ENDARCHIVE \(ESSCMD\)](#)

SSINVALIDTEXTDETECTION

Controls whether an Essbase error is generated when a spreadsheet user enters invalid text data into a cell that could possibly cause the user to misinterpret the data in the grid.

Syntax

```
SSINVALIDTEXTDETECTION TRUE | FALSE
```

- TRUE—An error message is displayed citing the invalid text and location, and saying to remove the text and retry.
- FALSE—Default value. No error message is displayed. The text that was entered is ignored.

Examples

```
SSINVALIDTEXTDETECTION TRUE
```

```
SSINVALIDTEXTDETECTION FALSE
```

SSLCIPHERSUITES

Defines one or more cipher suites to use for negotiating the security settings for a network connection using the SSL network protocol.

Syntax

```
SSLCIPHERSUITES ciphersuite_1[ciphersuite_2, ..., ciphersuite_6]
```

At least one cipher suite is required. A comma-delimited list of cipher suites, in order by preference, is supported. The first cipher suite in the list has the highest priority.

Description

You can change the default cipher suite.

1. SSL_RSA_WITH_RC4_128_MD5 (default)
2. SSL_RSA_WITH_RC4_128_SHA
3. SSL_RSA_WITH_3DES_EDE_CBC_MD5
4. SSL_RSA_WITH_DES_CBC_SHA
5. SSL_RSA_WITH_AES_128_CBC_SHA
6. SSL_RSA_WITH_AES_256_CBC_SHA

Note: For the highest level of security, reverse the order in which these cipher suites are listed.

Example

```
SSLCIPHERSUITES SSL_RSA_WITH_AES_128_CBC_SHA, SSL_RSA_WITH_DES_CBC_SHA
```

See Also

[“AGENTSECUREPORT” on page 388](#)

[“CLIENTPREFERREDMODE” on page 411](#)

[“ENABLECLEARMODE” on page 439](#)

[“ENABLESECUREMODE” on page 440](#)

[“NETSSLHANDSHAKETIMEOUT” on page 474](#)

[“WALLETPATH” on page 515](#)

For information on implementing SSL, see the *Oracle Hyperion Enterprise Performance Management System Security Administration Guide*.

SSLOGUNKNOWN

Controls whether Essbase logs error messages when it encounters an unknown member name during a spreadsheet operation.

Syntax

```
SSLOGUNKNOWN TRUE | FALSE
```

- TRUE—Essbase displays and logs an error message for each unknown member name that it encounters during a spreadsheet operation. The default is TRUE.
- FALSE—Essbase does not display error messages when it encounters an unknown member name nor does it log an error for each unknown member it encounters during a Spreadsheet operation.

Description

SSLOGUNKNOWN controls whether Essbase logs error messages when it encounters an unknown member name during a spreadsheet operation. It enables you to get a specific list of every unknown member name, or to repress error messages of this type.

Notes

SSLOGUNKNOWN creates an entry in the application log, *application_name.log*, in the application directory.

Example

```
SSLOGUNKNOWN TRUE
```

Essbase generates and logs an error message each time it encounters any number of unknown member names during a spreadsheet operation.

See Also

[“CLEARLOGFILE” on page 410](#)

[“TIMINGMESSAGES” on page 507](#)

SSOPTIMIZEDGRIDPROCESSING

Specifies whether optimized grid processing, which cuts the input grid into symmetric grids to create fewer symmetric queries, is enabled for spreadsheet 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 spreadsheet operations.
- FALSE—Disables optimized grid processing for spreadsheet operations.

The default value is FALSE.

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

Example

```
SSOPTIMIZEDGRIDPROCESSING TRUE
```

Enables optimized grid processing for spreadsheet operations on all applications and databases on Essbase Server.

SSPROCROWLIMIT

Controls the maximum number of spreadsheet rows Essbase processes on a Spreadsheet Add-in request.

Syntax

```
SSPROCROWLIMIT n
```

Where *n* is an integer value between 16,384 and 500,000, inclusive. The default value is 250,000.

Description

SSPROCROWLIMIT controls the maximum number of spreadsheet rows Essbase processes on a Spreadsheet Add-in user request. SSPROCROWLIMIT is in effect only for Spreadsheet Add-in 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, Analysis Services 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 Spreadsheet Add-in 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 spreadsheet-defined limits.
- If SSPROCROWLIMIT is exceeded, Essbase issues an error message and stops processing the request.
- If using Advanced Interpretation mode in the Spreadsheet Add-in, one cannot turn this setting off from the spreadsheet. The setting is not used in Freeform mode.

Example

```
SSPROCROWLIMIT 300000
```

SUPNA

Controls whether the Suppress #Missing Rows option in the spreadsheet 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 in the spreadsheet interface suppresses the display of cells for which a user has no access.
- OFF—The Suppress #Missing Rows option in the spreadsheet interface does not suppress the display of cells for which a user has no access. The default is OFF.

Description

The Suppress #Missing Rows option in the spreadsheet interface 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. The spreadsheet interface does not provide an equivalent.

Example

```
SUPNA ON
```

Essbase suppresses cells for which a user has no access.

```
SUPNA OFF
```

Essbase does not suppress cells for which a user has no access. These cells appear in the spreadsheet as #NoAccess. Rows of missing data are suppressed.

TARGETASOOPT

Potentially optimizes large queries (from the Spreadsheet Add-in, 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 the Spreadsheet Add-in, 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 disabled for the query. To enable TARGETASOOPT for the query, you must revalidate the partitions.

Example

```
TARGETASOOPT TRUE
```

See Also

[“TARGETTIMESERIESOPT” on page 507](#)

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
```

Example

```
TARGETTIMESERIESOPT
```

See Also

[“TARGETASOOPT” on page 506](#)

TIMINGMESSAGES

Controls whether Essbase logs the duration of each spreadsheet and report query in the application log.

Syntax

```
TIMINGMESSAGES TRUE | FALSE
```

- TRUE—Essbase logs these items:
 - The duration of all spreadsheet and report queries in the application log.
 - The log also records a timestamp of the query's execution.
 - Messages about dynamic calculator cache usage for each data retrieval.

The default setting is TRUE.

- FALSE—Essbase does not log these items:
 - The duration of all spreadsheet and report queries in the application log.
 - The log also records a timestamp of the query's execution.
 - Messages about dynamic calculator cache usage for each data retrieval.

If you have not created a .CFG file, or if you do not have this parameter specified in your .CFG file, Essbase automatically records and logs the duration of queries in the application log. You must set TIMINGMESSAGES to FALSE to disable this feature.

Description

TIMINGMESSAGES controls whether Essbase logs the duration of each spreadsheet and report query in the application log. Setting TIMINGMESSAGES to FALSE disables the logging of query durations in the application log. If the timing of queries is disabled, Essbase does not have to communicate with the operating system to get query start and finish times. As a result, query execution times may be improved in environments with many concurrent users. Disabling this parameter also decreases the size of the application log.

Example

```
TIMINGMESSAGES TRUE
```

Causes Essbase to time and log the duration of queries in the application log.

```
For example: [Thu Mar 19 14:55:32 1998]Local/Sample/Basic/admin/  
Info(1020055) Spreadsheet Extractor Elapsed Time : [0.078] seconds
```

```
TIMINGMESSAGES FALSE
```

disables the logging of query durations.

See Also

[“SSLOGUNKNOWN” on page 503](#)

TRANSACTIONLOGDATALOADARCHIVE

Specifies the type of data to archive when logging transactions. By default, Essbase archives only data load and rules files for client data loads.

During transaction logging, Essbase creates archive copies of data load and rules files in the following directory:

```
ARBORPATH/app/appname/dbname/Replay
```

These files are then used during the replay of a logged transaction.

To enable transaction logging and replay, use the TRANSACTIONLOGLOCATION configuration setting.

Transaction logging and replay, used with the automated backup and restore feature, facilitates recovery of an Essbase block storage database. Transaction logging and replay does not apply to aggregate storage databases. See the *Oracle Hyperion Enterprise Performance Management System Backup and Recovery Guide*.

Syntax

```
TRANSACTIONLOGDATALOADARCHIVE [appname [dbname]] [OPTION]
```

- *appname*—Optional. Specifies the application for which to archive the data and rules associated with logged transactions.

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

- *dbname*—Optional. Specifies the database, in the application specified by *appname*, for which to archive the data and rules associated with logged transactions.

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

- *OPTION*—Valid values are as follows:
 - CLIENT: (Default) Archives data load and rules files for client data loads.
 - SERVER: Archives data load and rules files on the server and SQL-server data loads.

Caution! Server data loads are replayed using the data load and rules files that are archived on the server in the `Replay` directory. Do not rename these files. Also, if the contents of the data load and rules files are changed before the replay operation, the modified data is used during replay. Therefore, the data in the recovered database will not be the same as the original data.

- SERVER_CLIENT: Archives server and client data.
- NONE: No data is archived.

If you select NONE and use client data, Essbase cannot replay the data load. In this case, to recover transactions, you must manually load the client data before you replay the remaining transactions.

If you use server or SQL data, and the data and rules files are not archived in the `Replay` directory (for example, you did not use the SERVER or SERVER_CLIENT option), Essbase replays the data that is currently in the data source, which may or may not be the data that was originally loaded.

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

Example

```
TRANSACTIONLOGDATALOADARCHIVE SERVER_CLIENT
```

Archives server and client data for all databases on Essbase Server.

See Also

[“TRANSACTIONLOGLOCATION” on page 510](#) configuration setting

[Query Database](#) MaxL statement

[Alter Database](#) MaxL statement

TRANSACTIONLOGLOCATION

Specifies whether to enable write transaction logging and specifies an existing directory on Essbase Server for the transaction log store.

Transaction logging and replay, used with the automated backup and restore feature, facilitates recovery of an Essbase block storage database. Transaction logging and replay does not apply to aggregate storage databases. See the *Oracle Hyperion Enterprise Performance Management System Backup and Recovery Guide*.

Syntax

```
TRANSACTIONLOGLOCATION [appname [dbname]] LOGLOCATION NATIVE ENABLE | DISABLE
```

- *appname*—Optional. Specifies the application for which transaction logging and replay is to be enabled.

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

- *dbname*—Optional. Specifies the database, in the application specified by *appname*, for which transaction logging and replay is to be enabled.

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

- *LOGLOCATION*—Specifies the directory in which the transaction log store is written.

Oracle recommends specifying multiple log locations.

Note: Oracle recommends using a physical disk other than the disk on which the *ARBORPATH* directory or disk volumes reside.

- *NATIVE*—Specifies a reserved field.

Note: Do not change the *NATIVE* value.

- *ENABLE* | *DISABLE*—Specifies whether to enable or disable transaction logging.

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

Description

You can use multiple TRANSACTIONLOGLOCATION statements to enable transaction logging at a more global level and, at the same time, disable logging at a more granular level. In the `essbase.cfg` file, the more global enabling statement must precede the more granular disabling statement for the override to take effect.

Note: If transaction logging is enabled for an application or database that you later rename or copy, you must enable logging for the renamed or copied application or database and you must use the same path that is specified in the TRANSACTIONLOGLOCATION setting.

Example

```
TRANSACTIONLOGLOCATION Sample C:\Hyperion\trlog NATIVE ENABLE
```

Enables transaction logging for all databases associated with the Sample application and writes the log store to the `trlog` directory.

```
TRANSACTIONLOGLOCATION Hyperion/trlog NATIVE ENABLE  
TRANSACTIONLOGLOCATION Sample Hyperion/trlog NATIVE DISABLE
```

The first statement enables transaction logging for all applications and their associated databases on Essbase Server; the second statement disables transaction logging for all databases associated with a specific application (Sample).

```
TRANSACTIONLOGLOCATION Sample Hyperion/trlog/Sample NATIVE ENABLE  
TRANSACTIONLOGLOCATION Sample Basic Hyperion/trlog/Sample NATIVE DISABLE
```

The first statement enables transaction logging at the application level (Sample); the second statement disables transaction logging for a specific database (Basic) in the application.

See Also

[“TRANSACTIONLOGDATALOADARCHIVE” on page 508](#) configuration setting

[Query Database MaxL statement](#)

[Alter Database MaxL statement](#)

TRIGMAXMEMSIZE

Specifies the maximum amount of memory that Essbase can allocate to the triggers feature.

Syntax

```
TRIGMAXMEMSIZE [application [database]] memsize
```

- *application*—Optional. Sets the available memory cache for all databases in the specified application.
- *database*—Optional. Sets the available memory cache for the specified database. If you specify a database, you must specify the application that contains it.

- *memsize*—Available memory cache size (in bytes). Default: 4096 bytes. Minimum: 4096 bytes. Maximum: 8388608 bytes (8MB). Setting *memsize* to zero (0), or a negative value, disables all triggers.

Description

TRIGMAXMEMSIZE specifies the maximum amount of memory available to the Essbase triggers feature. The triggers feature lets you efficiently monitor data changes in a database. If data breaks the rules that you have specified, Essbase logs the information in a file or sends an email alert.

For more information about triggers, see the *Oracle Essbase Database Administrator's Guide*. For information about MaxL triggers statements, see the *Oracle Essbase Technical Reference*.

Notes

- You must specify the memory in bytes. If you specify a size greater than the maximum of 8388608 bytes, Essbase automatically sets the size to 8388608 bytes.

Example

```
TRIGMAXMEMSIZE 12288
```

sets the maximum memory cache for the triggers feature to 12288 bytes (12K). The setting applies to all applications and databases on the Essbase Server.

See Also

[create trigger](#) (MaxL statement)

[display trigger](#) (MaxL statement)

[alter trigger](#) (MaxL statement)

[drop trigger](#) (MaxL statement)

UNICODEAGENTLOG

Specifies whether the Essbase Server log (*essbase.log*) is written in UTF-8 encoding or according to the locale of the system. The system locale is defined by the optional `ESSLANG` variable or, if `ESSLANG` is not specified, by the computer operating system.

Syntax

```
UNICODEAGENTLOG NONUNICODE | UTF-8
```

- `NONUNICODE`—Encodes the Essbase Server log according to the locale of the system. The default is `NONUNICODE`.
- `UTF-8`—Encodes the Essbase Server log in UTF-8.

Description

By default, the Essbase Server log is encoded according to the locale of the system. Unicode-mode object names such as application and database names from different locales could be displayed in unrecognizable characters in the Essbase Server log. To avoid this problem, if you implement Unicode-mode applications, change the Essbase Server log to UTF-8 encoding. In UTF-8 encoding, a UTF-8-capable viewer or editor displays the characters accurately.

Notes

- To have the Essbase Server log written in UTF-8 encoding, backup or delete `essbase.log`, set `UNICODEAGENTLOG` to UTF-8, and restart Essbase Server. All entries to the log will be written in UTF-8 encoding. See the Example section.
- To have the Essbase Server log written in non-Unicode encoding, backup or delete `essbase.log`, change the `UNICODEAGENTLOG` setting to `NONUNICODE` (or remove the `UNICODEAGENTLOG` configuration setting), and restart Essbase Server. All entries to the log will be encoded according to the system locale.
- Any parameter value other than UTF-8 is interpreted as `NONUNICODE`. The case is not important.
- For more information about the Essbase implementation of Unicode, see the *Oracle Essbase Database Administrator's Guide*

Example

```
UNICODEAGENTLOG UTF-8
```

Causes the cleared Essbase Server log to be written in UTF-8 encoding when Essbase Server is restarted.

UPDATECALC

Controls whether Intelligent Calculation is turned on or off by default.

This setting does not apply to aggregate storage databases.

Syntax

```
UPDATECALC TRUE | FALSE
```

- `TRUE`—Intelligent Calculation is turned on. Essbase calculates only updated blocks and their dependent parents.
- `FALSE`—Intelligent Calculation is turned off. Essbase calculates all data blocks, regardless of whether they have been updated.

Description

`UPDATECALC` specifies whether Intelligent Calculation is turned on or off by default.

If required during a calculation, you can override this default setting and turn Intelligent Calculation on and off using the `SET UPDATECALC` command in a calculation script.

Using Intelligent Calculation, Essbase calculates only updated data blocks and their dependent parents. Therefore, the calculation is very efficient.

Notes

For more information on Intelligent Calculation, see the *Oracle Essbase Database Administrator's Guide*

Example

```
UPDATECALC TRUE  
UPDATECALC FALSE
```

In `essbase.cfg`, a parameter is not followed by a semicolon; in a calculation script, a parameter must be followed by a semicolon.

See Also

[SET CLEARUPDATESTATUS](#) (calculation command)

[SET UPDATECALC](#) (calculation command)

VLBREPORT

Enables Essbase to dynamically determine the retrieval buffer size, between 100 KB and 10 MB, for retrievals from databases without Dynamic Calc, attribute, or Dynamic Time Series members.

This setting does not apply to aggregate storage databases.

Syntax

```
VLBREPORT TRUE | FALSE
```

- **TRUE**—Essbase dynamically determines the size of the retrieval buffer for outlines that qualify.
- **FALSE**—Essbase does not dynamically determine the size of the retrieval buffer. The default setting is FALSE.

Description

Retrieved data is temporarily stored in the retrieval buffer. The size of the buffer can significantly affect retrieval performance. If you have a database which has a very large block size and you retrieve a large percentage of cells from each block across several blocks, consider setting the VLBREPORT option to TRUE. When this is done, Essbase dynamically determines the optimum buffer size between 100 KB and 10 MB, overriding the current retrieval buffer size setting.

Turning on VLBREPORT optimization can improve performance for concurrent and serial queries. Since queries can be completed faster, this optimization also enables more users to perform their queries at the same time.

Notes

- The VLBREPORT configuration setting applies only to databases that contain no Dynamic Calc, attribute, or Dynamic Time Series members.
- The retrieval buffer size is a database property which you can change using the MaxL `alter database` statement, the ESSCMD `SETDBSTATEITEM`, or Administration Services Console.
- For more information about setting the retrieval buffer size, see the *Oracle Essbase Database Administrator's Guide*.

Example

```
VLBREPORT TRUE
```

See Also

[alter database](#) (MaxL statement)

[SETDBSTATEITEM](#) (ESSCMD)

WALLETPATH

Specifies the path to the Oracle Wallet for Essbase Agent, Server, or Client for SSL communication.

Syntax

```
WALLETPATH path
```

Where *path* is a fully-qualified path that contains less than 1,024 characters. The default path is `ARBORPATH/bin/wallet`.

Description

To set up Oracle Wallet, you need the Oracle public key infrastructure (PKI) command line tool, `orapki`. You use the `orapki` utility to manage public key infrastructure elements such as wallets and certificate revocation lists.

Notes

For information about implementing SSL and setting up Oracle Wallet, see the *Oracle Hyperion Enterprise Performance Management System Security Administration Guide*.

Example

```
WALLETPATH /usr/local/wallet/agent
```

See Also

[“AGENTSECUREPORT”](#) on page 388

[“CLIENTPREFERREDMODE”](#) on page 411

[“ENABLECLEARMODE” on page 439](#)

[“ENABLESECUREMODE” on page 440](#)

[“NETSSLHANDSHAKETIMEOUT” on page 474](#)

[“SSLCIPHERSUITES” on page 502](#)

XOLAPENABLEHEURISTICS

Governs the extent to which the SQL for an application is to be optimized.

Syntax

```
XOLAPENABLEHEURISTICS [appname] TRUE | FALSE
```

- *appname* (optional)—Application name. If you do not specify an application, the setting (TRUE or FALSE) will apply to all XOLAP-enabled cubes.
- TRUE—SQL optimization is performed based upon the number of members in the levels of an XOLAP-enabled cube.
- FALSE—SQL optimization is not performed. FALSE is the default value.

Description

The XOLAPENABLEHEURISTICS setting governs the extent to which the SQL for an application is to be optimized. The optimization is based upon the number of members in the levels of the XOLAP-enabled cube.

Example

```
XXOLAPENABLEHEURISTICS my_app TRUE
```

See Also

[“XOLAPSQLIDLEPERIOD” on page 518](#)

[“XOLAPMAXNUMCONNECTION” on page 516](#)

[“XOLAPSCHEMAVERIFICATION” on page 517](#)

XOLAPMAXNUMCONNECTION

Specifies the maximum number of active connections that Essbase will maintain in the Global Connection Pool.

Syntax

```
XOLAPMAXNUMCONNECTION [appname] [value]
```

- *appname* (optional)—Application name. If you do not specify an application, the specified maximum number of connections will apply to all XOLAP-enabled cubes.

- *value*—The maximum number of active connections that Essbase will maintain in the Global Connection Pool. The default is 25 connections.

Description

The XOLAPMAXNUMCONNECTION setting specifies the maximum number of active connections that Essbase will maintain in the Global Connection Pool. The term *active connection* denotes an open connection to the RDBMS. If, during a query session, there is a need for more connections than are available in the Global Connection Pool, then they are created and deleted after the query session is finished.

Example

```
XOLAPMAXNUMCONNECTION my_app 16
```

See Also

[“XOLAPSQLIDLEPERIOD” on page 518](#)

[“XOLAPSCHEMAVERIFICATION” on page 517](#)

[“XOLAPENABLEHEURISTICS” on page 516](#)

XOLAPSCHEMAVERIFICATION

Determines whether the XOLAP schema supplied for an application is validated against the underlying RDBMS.

Syntax

```
XOLAPSCHEMAVERIFICATION [appname] TRUE | FALSE
```

- *appname* (optional)—Application name. If you do not specify an application, the setting (TRUE or FALSE) will apply to all XOLAP-enabled cubes.
- TRUE—SQL queries are issued to validate the relational data provided in the XML file.
- FALSE—SQL queries are not issued, and the relational data in the XML file is not validated. FALSE is the default value.

Description

The XOLAPSCHEMAVERIFICATION setting determines whether the XOLAP schema supplied for an application is validated against the underlying RDBMS. The validation occurs as SQL queries are run against the relational data in the XML file.

Example

```
XOLAPSCHEMAVERIFICATION my_app TRUE
```

See Also

[“XOLAPSQLIDLEPERIOD” on page 518](#)

[“XOLAPMAXNUMCONNECTION” on page 516](#)

[“XOLAPENABLEHEURISTICS” on page 516](#)

XOLAPSQLIDLEPERIOD

Specifies the maximum number of minutes a connection can remain idle before it is tested.

Syntax

```
XOLAPSQLIDLEPERIOD [appname] value
```

- *appname* (optional)—Application name. If you do not specify an application, the specified maximum number of minutes will apply to all XOLAP-enabled cubes.
- *value*—The maximum number of minutes a connection can remain idle before it is automatically tested prior to being used by an application. The default is 30 minutes.

Description

The XOLAPSQLIDLEPERIOD setting specifies the maximum number of minutes a connection can remain idle before it is automatically tested prior to being used by an application.

Example

```
XOLAPSQLIDLEPERIOD my_app 20
```

See Also

[“XOLAPMAXNUMCONNECTION” on page 516](#)

[“XOLAPSCHEMAVERIFICATION” on page 517](#)

[“XOLAPENABLEHEURISTICS” on page 516](#)

5

ESSCMD Commands

In This Chapter

ESSCMD Overview	519
ESSCMD Getting Started	519
ESSCMD Syntax Guidelines	520
ESSCMD Batch Processing	522
ESSCMD Interactive Mode	526
ESSCMD Command Groups	528
ESSCMD Command Reference	534

ESSCMD Overview

ESSCMD is a command-line interface that performs operations interactively or through a batch or script file. You can execute Essbase operations at the command line, in either batch or interactive mode:

- **Interactive mode**—Enables you to interactively enter commands at the ESSCMD command line and receive responses. Interactive mode is convenient for short operations that require few commands, checking for information on the fly, and error checking.; see [“ESSCMD Interactive Mode” on page 526](#).
- **Batch-processing mode**—Enables you to automate your routine Essbase maintenance and diagnostic tasks. You can write a script or batch file and run it from the command line. Batch processing mode is convenient if you frequently use a particular series of commands, or if your task requires many commands; see [“ESSCMD Batch Processing” on page 522](#).

ESSCMD operates independently of any other Essbase client interface, including Administration Services, Spreadsheet Add-in, or custom-built application programs.

Because ESSCMD supports multiple login instances to Essbase Server, you can access multiple databases in one session. Even when you log in to multiple databases, you use only one port on your Essbase Server license.

ESSCMD Getting Started

Before you start ESSCMD, make sure that the following items are properly installed and running:

- Essbase Server

- Communications protocol (TCP/IP)

Starting ESSCMD

The Essbase Server installation places the `ESSCMD.EXE` and `ESSCMD.HLP` files (`ESSCMD` and `esscmd.hlp` on UNIX platforms) in the `bin` directory.

- To start ESSCMD, enter `ESSCMD` at the operating system command prompt.

ESSCMD runs within the operating system command prompt.

Once you start the application, a command prompt like this one appears:

```
::: [#] ->
```

where # is the value of the active login instance. Each subsequent, successful login increments this value by one. When you start ESSCMD, the instance number is zero (0).

Canceling ESSCMD Operations

When running ESSCMD, you can cancel an asynchronous operation, such as a calculation, export, or restructure operation, by pressing and holding the Esc key until ESSCMD responds.

Quitting ESSCMD

- To quit ESSCMD, enter `EXIT` at the prompt and press Enter.

ESSCMD disconnects from Essbase Server and returns to the operating system command prompt.

ESSCMD Syntax Guidelines

There are some differences between ESSCMD's interactive and batch processing modes in the requirements for quotation marks and the semicolon statement terminator. Use the guidelines in this section when creating script or batch files.

Case-sensitivity varies by operating system:

- Windows is not case-sensitive. You can enter ESSCMD commands and file-names in uppercase or lowercase letters, or in any combination of the two.
- UNIX is case-sensitive. You must enter file names in the correct case or UNIX does not recognize them. However, you can enter ESSCMD command names and parameters in uppercase or lowercase.

Quotation Marks in ESSCMD

Double quotation marks (" ") enclose character parameters and responses to commands.

- In interactive ESSCMD, using double quotation marks is optional. Be sure to use them when a parameter has an embedded space; for example,

```
CALC "Calc All;";
```

- In an ESSCMD script file, always enclose all character parameters and responses to commands in double quotation marks; for example,

```
LOGIN "Local" "TomT" "Password";
```

- You do not have to enclose numeric parameters and responses in quotation marks.
- You cannot place quotation marks within quotation marks.

ESSCMD Semicolon Statement Terminator

The ; (semicolon) statement terminator signals the end of a command; for example,

```
SELECT "SAMPLE" "BASIC";
```

- In interactive ESSCMD, pressing the Enter key signals ESSCMD that the command is complete. The statement terminator is optional.
- In an ESSCMD script file, you should use the terminator, even though it is optional, if a command has many parameters. This is especially important in order to signal the end of the parameter list if some of the parameters are optional.
- If you omit some optional parameters and do not use a semicolon to end the list, ESSCMD looks for the remaining values in the next command in the file, leading to unpredictable results.

The [SETAPPSTATE](#) and [SETDBSTATE](#) commands are examples of commands which you should terminate with ; to prevent any confusion in processing.

Note: All syntax examples in this documentation use quotation marks and semicolon terminators.

Referencing Files

Some commands require that you precede object or file names with a numeric parameter, from 1 to 4, that tells ESSCMD where to look for the object or file. The parameter directs ESSCMD to look for files in other applications, databases, or systems.

The following table lists each value for the numeric parameter (numeric), the file location to which it applies, and the information that ESSCMD requests when you use each parameter setting. appName is the application name and dbName is the database name.

Numeric	File	ESSCMD prompts for:
1	Local or client-based file	Windows: Files in the <code>\ARBORPATH\CLIENT\appName\dbName</code> directory. UNIX: Files in the <code>\$ARBORPATH/client/appName/dbName</code> directory.
2	Remote or server-based file	Windows: Files in the <code>\ARBORPATH\APP\appName\dbName</code> directory. UNIX: Files in the <code>\$ARBORPATH/app/appName/dbName</code> directory.
3	File	Fully-qualified path to the file, unless file is in the current ESSCMD directory.
4	SQL table	Full network and database information for the SQL table.

For example, the `LOADDATA` command can load a data file that resides on the client computer or the Essbase Server computer. The command requires the numeric parameter to tell ESSCMD where to look for the data file. This example causes ESSCMD to prompt for the fully-qualified path name of the file to load:

```
LOADDATA 3
```

File extensions are usually optional in both interactive and batch processing modes, except when using commands that require a numeric parameter that indicates the location of files:

- If you use file option 3 (File), you must enter the file extension in both interactive and batch processing modes.
- If the object is in the directory from which you started ESSCMD, you do not need to enter a path.

ESSCMD Batch Processing

If you use a series of commands frequently or you must enter many commands to complete a task, consider script or batch file automation.

- You can run a script file containing ESSCMD commands from the operating system command line or from an operating system batch file. A script has a `.SCR` extension.
- A batch file is an operating system file that calls multiple ESSCMD scripts, and may include operating system commands. You can use a batch file to run multiple sessions of ESSCMD. On Windows systems, batch files have `.BAT` extensions.

Note: On UNIX, a batch file is a shell script. A shell script usually has the file extension `.sh` (Bourne or Korn shell) or `.csh` (C shell).

When you run a script or batch file, ESSCMD executes the commands in order until the end of the file.

Writing Script Files

Each script must be a complete ESSCMD session, with login, application and database selection, logout, and termination commands.

To create a script:

1. Enter ESSCMD commands in a text editor.
2. Save the file with the `.SCR` extension.

For example, the following script file, `TEST.SCR`, was created in Notepad:

```
LOGIN "LOCAL" "TOMT" "PASSWORD";  
SELECT "SAMPLE" "BASIC";  
GETDBSTATE  
EXIT;
```

When run from the operating system command line, this script logs TomT into the Local server, selects the Sample application and Basic database, gets database statistics, and quits ESSCMD.

Running Script Files

Enter the following command at the operating system prompt:

```
ESSCMD scriptFileName.SCR
```

Replace *scriptFileName* with the name of the script file. For example, type the following if the script file is in the current directory:

```
ESSCMD TEST.SCR
```

If the script file is in another directory, include the path. For example:

```
ESSCMD C:\WORK\SCRIPTS\TEST.SCR (absolute path on Windows)
```

or

```
ESSCMD ..\SCRIPTS\TEST.SCR (relative path on Windows)
```

Handling Command Errors in a Script File

ESSCMD provides error checking and handling. You can check for errors and, if necessary, branch to an appropriate response.

After each ESSCMD command is executed, a number is stored in an internal buffer. If the command executes successfully, 0 is returned to the buffer; if the command is unsuccessful, the error number is stored in the buffer. Unsuccessful execution is called non-zero status.

ESSCMD provides the following error-handling commands:

- `IFERROR` checks the previously executed command for a non-zero (failure) return status. If the status is not zero, processing skips all subsequent commands and jumps to resume at a user-specified point in the file.
- The script file can branch to an error-handling routine or the end of the file.

- RESETSTATUS reverts all saved status values to 0 (zero) in preparation for more status checking.
- GOTO forces unconditional branching to a user-specified point in the file, whether or not an error occurred.

In this LOAD.SCR example, if a command does not execute successfully, ESSCMD branches to the end of the file to avoid completing other operations.

```

LOGIN "local" "User1" "password" "Sample" "Basic";
LOADDATA 2 "calcdat";
IFERROR "Error";
CALC "Calc All;";
IFERROR "Error";
RUNREPT 2 "Myreport";
IFERROR "Error";
[possible other commands]
EXIT;

:Error

EXIT;

```

Note: You can use the OUTPUT command to log errors.

Sample Script Files

The following script files demonstrate common batch operations. All samples are based on the Sample Basic database. The scripts for these examples are available in `\ARBORPATH\APP\SAMPLE\BASIC`. On UNIX systems, the examples are available from `/home/$ARBORPATH/app/Sample/Basic`.

Importing and Calculating a Data Sample File

Suppose you need a file that:

- Logs in to Essbase Server.
- Selects an application and database.
- Prevents other users from logging on and making changes to the database.
- Imports data from a text file.
- Calculates the database.
- Exits ESSCMD.

The following script file does the job:

```

LOGIN "Poplar" "TomT" "Password";
SELECT "Sample" "Basic";
DISABLELOGIN;
IMPORT 2 "ACTUALS" 4 "Y" 2 "ACTUAL" "N";
CALCDEFAULT;

```

```
ENABLELOGIN;  
EXIT;
```

On Windows, this script file, `sample1.scr`, is available in `\ARBORPATH\APP\SAMPLE\BASIC`. On UNIX, `sample.scr` is in `/$ARBORPATH/app/Sample/Basic`.

Updating a SQL Script, Importing, and Calculating a Data Sample File

Suppose you need a script file that:

- Logs in to Essbase Server.
- Selects an application and database.
- Prevents other users from logging on and making changes to the database.
- Updates the outline from an SQL data source.
- Imports data from SQL.
- Calculates the database.
- Exits ESSCMD.

The following script file does the job:

```
LOGIN "Poplar" "TomT" "Password";  
SELECT "Sample" "Basic";  
DISABLELOGIN;  
BUILDDIM 2 "PRODRUL" 4 "PRODTBL" 4  
"  
PROD.ERR  
"  
;  
IMPORT 4 "TOMT" "PASSWORD" 2 "ACTUAL" "N";  
CALCDEFAULT;  
EXIT;
```

On Windows, this script file, `sample2.scr`, is available in `\ARBORPATH\APP\SAMPLE\BASIC`. On UNIX, it is in the `/$ARBORPATH/app/Sample/Basic` directory.

Writing Batch Files

You can write a batch file that runs one or more report scripts, and includes operating system commands. See your operating system instructions to learn the syntax for writing batch files.

Handling Command Errors in Batch Files

For the operating system batch file, you can use ESSCMD command return values to control the flow of scripts that the batch file executes.

An ESSCMD program returns an integer value upon exiting. This value represents the status of the last executed command. You can set up your batch file to test for this value, and if the test

fails, branch to a response. For information about handling errors in script files, see [Handling Command Errors in a Script File](#).

For example, a batch file could contain three scripts: an ESSCMD batch file that loads data, a calc script, and a report script. If the load batch file fails, the calculations and reporting also fail. In this case, it would be best to stop the batch file and correct the error. If your batch file tests for the return value of the load process, and this return value indicates failure, the batch file can jump to the end of the file and stop or execute some other error-handling procedure, rather than attempting to calculate data that did not load.

The following example shows a Windows operating system batch file and the contents of one of the ESSCMD scripts it runs, LOAD.SCR. Because error-checking requirements vary, the syntax in this example may not correspond to that of your operating system. See your operating system documentation for error checking in batch files.

```
ESSCMD LOAD.SCR
If not %errorlevel%==goto Error
ESSCMD CALC.SCR
If not %errorlevel%==goto Error
ESSCMD REPORT.SCR
If not %errorlevel%==goto Error
Echo All operations completed successfully
EXIT

:Error
Echo There was a problem running the script
```

Sample Script: Scheduling Report Printing

Suppose you need a file that:

- Logs in to Essbase Server.
- Selects an application and database.
- Assigns reports that output to files for later printing.
- Exits ESSCMD.

The following script file does the job:

```
LOGIN "Poplar" "TomT" "Password";
SELECT "Sample" "Basic";
RUNREPT 2 "REP1" "REP1.OUT";
RUNREPT 2 "REP2" "REP2.OUT";
RUNREPT 2 "REP3" "REP3.OUT";
EXIT;
```

On Windows, SAMPLE3.SCR is available in `\ARBORPATH\APP\SAMPLE\BASIC`. On UNIX, SAMPLE3.SCR is in `/$ARBORPATH/app/Sample/Basic`.

ESSCMD Interactive Mode

In interactive mode, you enter commands and respond to prompts. This is useful when you are performing simple tasks that require few commands. If you are performing more complex tasks

that require many commands, consider creating a script file or batch file; see “[ESSCMD Batch Processing](#)” on page 522.

For syntax conventions when working in interactive mode, see “[ESSCMD Syntax Guidelines](#)” on page 520.

Logging On to Essbase Server

After starting ESSCMD, you must connect to Essbase Server so that you can enter commands.

➤ To log on to Essbase Server:

- 1 At the ESSCMD prompt, log in to Essbase Server with the LOGIN command.
- 2 Enter the host name for Essbase Server. When you connect from the server console, the server name depends on your network setup. For example, the name could be LOCAL.
- 3 Enter your user name.
- 4 Enter your password.

The ESSCMD prompt appears as follows:

```
local:::u
serName
[1]->
```

where *userName* is your login name.

You can enter any valid ESSCMD command (see “[ESSCMD Command Reference](#)” on page 534).

Note: To load an application into memory and select a database, use the SELECT command.

The ESSCMD prompt appears as follows:

```
local:appName:dbName:userName[1]->
```

where:

- *appName* is the name of the application.
- *dbName* is the name of the database to which you are connected.

Entering Commands

➤ To enter commands in interactive mode, select one method:

- Type the command and press **Enter**.

ESSCMD prompts you for each of the command parameters. For example, the SELECT command has two parameters, as shown in the command syntax:

```
SELECT "appName" "dbName";
```

If you enter only `SELECT` and press `Enter`, ESSCMD prompts you for the first parameter, the application name (`appName`). After you enter the application name and press `Enter`, ESSCMD prompts you for the database name (`dbName`).

- Type the commands and all parameters, then press **Enter**.

Using `SELECT` as the example, you would type:

```
SELECT "Sample" "Basic";
```

Whichever method you use, the interactive prompt now reflects the application and database names. For example, the following prompt tells you that the `Sample` application and `Basic` database are selected:

```
local:Sample:Basic:User[1]->
```

In this case, you can enter other commands without the application or database name parameters that it normally requires.

Canceling Operations

While ESSCMD is running, you can cancel an asynchronous operation, such as a calculation, export, or restructure operation, by pressing and holding the `Esc` key until ESSCMD responds.

Warning: Do not pause or suspend your system (for example, by pressing `Ctrl-S`) while Essbase Server is processing a command. Pausing the system may prevent Essbase Server from correctly completing the command.

ESSCMD Command Groups

This topics in this section list ESSCMD commands, grouped by command type.

Using ESSCMD

Use these commands to log in and out of ESSCMD, view a list of commands, pause an ESSCMD script, and redirect command output:

- `LOGIN`
- `LOGOUT`
- `LISTLOGINS`
- `SETLOGIN`
- `SLEEP`
- `SELECT`
- `EXIT`

Application and Database Administration

Use these commands to perform database administration, and get information about applications and databases:

- COPYAPP
- COPYDB
- CREATEAPP
- CREATEDB
- DELETEDAPP
- DELETEDB
- GETAPPACTIVE
- GETAPPINFO
- GETAPPSTATE
- GETDBACTIVE
- GETDBINFO
- GETDBSTATE
- GETDBSTATS
- GETVERSION
- LISTAPP
- LISTDB
- LISTFILES
- LOADAPP
- LOADDB
- RENAMEAPP
- RENAMEDB
- SETAPPSTATE
- SETDBSTATE
- SETDBSTATEITEM
- SHUTDOWNSERVER
- UNLOADAPP
- UNLOADDB

User and Group Security

Use these commands to perform user and group administration:

- DISABLELOGIN

- [ENABLELOGIN](#)
- [LOGOUTUSER](#)
- [LOGOUTALLUSERS](#)
- [CREATEUSER](#)
- [DELETEUSER](#)
- [RENAMEUSER](#)
- [LISTUSERS](#)
- [GETUSERINFO](#)
- [SETPASSWORD](#)
- [CREATEGROUP](#)
- [DELETEGROUP](#)
- [LISTGROUPS](#)
- [ADDUSER](#)
- [REMOVEUSER](#)
- [LISTGROUPUSERS](#)

Security Filters and Locks

Use these commands to list, copy and rename security filters, and to view and remove database locks:

- [COPYFILTER](#)
- [LISTFILTERS](#)
- [LISTLOCKS](#)
- [REMOVELOCKS](#)
- [RENAMEFILTER](#)

Database Objects

Use these commands to list database objects and their lock statuses, copy and rename database objects, and to view and remove URLs, cell notes, or partitions linked to the database:

- [LISTOBJECTS](#)
- [COPYOBJECT](#)
- [RENAMEOBJECT](#)
- [UNLOCKOBJECT](#)
- [LISTLINKEDOBJECTS](#)
- [PURGELINKEDOBJECTS](#)

Outline and Attribute Information

Use these commands to view member information, attribute information, current attribute naming specifications for the database, and view outline paging information:

- [GETMBRINFO](#)
- [GETMEMBERS](#)
- [GETATTRINFO](#)
- [GETATTRIBUTESPECS](#)

Dimension Building

You can build multiple dimensions incrementally, with or without automatic restructuring after the dimension build is complete.

Use these commands to build one or more dimensions from data files or SQL sources:

- [BUILDDIM](#)
- [INCBUILDDIM](#)
- [BEGININCBUILDDIM](#)
- [ENDINCBUILDDIM](#)

Data Loading, Clearing, and Exporting

Use these commands to load data files or individual records, to clear all data from the database, or to export and import data to and from a text file:

- [LOADDATA](#)
- [UPDATE](#)
- [UPDATEFILE](#)
- [RESETDB](#)
- [EXPORT](#)
- [PAREXPORT](#)
- [IMPORT](#)

Calculating

Use these commands to run calc scripts, execute one or more calc strings, run or change the default calculation, and view information about calc strings associated with members:

- [CALC](#)
- [CALCDEFAULT](#)
- [CALCLINE](#)

- [RUNCALC](#)
- [GETMBRCALC](#)
- [GETDEFAULTCALC](#)
- [SETDEFAULTCALC](#)
- [SETDEFAULTCALCFILE](#)

Reporting

Use these commands to run report scripts and execute one or more report strings:

- [RUNREPT](#)
- [REPORT](#)
- [REPORTLINE](#)

Partitioning

To produce a text file of the distributed database's partition mapping tables, use the [PRINTPARTITIONDEFFILE](#) command.

To replicate all data cells, or only updated data cells, in a replicated partition, use these commands:

- [GETALLREPLCELLS](#)
- [PUTALLREPLCELLS](#)
- [GETUPDATEDREPLCELLS](#)
- [PUTUPDATEDREPLCELLS](#)

Use "GET" commands to replicate cells from source to target, when you are working from the computer hosting the target database.

Use "PUT" commands to replicate cells from source to target, when you are working from the computer hosting the source database.

Outline Synchronization

Outline synchronization commands utilize an outline change file (.CHG) to track changes made on the source outline, apply those changes to the target outline, and synchronize time stamps in the partition definition files.

Use these commands to keep the target database outline synchronized with changes made to the source database outline:

- [GETPARTITIONOTLCHANGES](#)
- [APPLYOTLCHANGEFILE](#)

- [PURGEOTLCHANGEFILE](#)
- [RESETOTLCHANGETIME](#)

Error and Log Handling

Use these commands for conditional and unconditional error branching in ESSCMD scripts, redirection of process information, specifying what kind of messages are displayed, and clearing the application log file:

- [RESETSTATUS](#)
- [SETMSGLEVEL](#)
- [GOTO](#)
- [IFERROR](#)
- [OUTPUT](#)
- [DELETELOG](#)

Currency Conversion Information

Use these commands to get information about the currency database linked to the currently selected database:

- [GETCRDB](#)
- [GETCRDBINFO](#)
- [GETCRRATE](#)
- [GETCRTYPE](#)

Location Aliases

Location aliases are names representing host-application-database-user name-password combinations.

Use these commands to manage location aliases in a distributed Essbase environment:

- [CREATELOCATION](#)
- [DELETELOCATION](#)
- [LISTLOCATIONS](#)

Substitution Variables

Substitution variables are placeholders for information that changes regularly. Use them in calc scripts, report scripts, and the Spreadsheet Add-in.

Use these commands to manage substitution variables:

- [CREATEVARIABLE](#)
- [DELETEVARIABLE](#)
- [LISTVARIABLES](#)
- [UPDATEVARIABLE](#)

Aliases

Alias tables contain a listing of member names and their alternate names, or aliases. Create alias tables using Administration Services.

Use these commands to manage and display the contents of alias tables for a database:

- [LISTALIASES](#)
- [SETALIAS](#)
- [LOADALIAS](#)
- [UNLOADALIAS](#)
- [DISPLAYALIAS](#)

Integrity, Performance

Use these commands to get and reset performance statistics, and check for data integrity:

- [GETPERFSTATS](#)
- [RESETPERFSTATS](#)
- [VALIDATE](#)

Backing Up

Use these commands to place a database in read-only mode in preparation for archiving, and to restore the database to read-write mode after archiving is complete:

- [BEGINARCHIVE](#)
- [ENDARCHIVE](#)

ESSCMD Command Reference

Consult the Contents pane for a categorical list of ESSCMD commands.

[ADDUSER](#)

[GETCRTYPE](#)

[PRINTPARTITIONDEFFILE](#)

[APPLYOTLCHANGEFILE](#)

[GETDBACTIVE](#)

[PURGELINKEDOBJECTS](#)

[APPLYOTLCHANGEFILEEX](#)

[GETDBINFO](#)

[PURGEOTLCHANGEFILE](#)

ADDUSER	GETCRTYPE	PRINTPARTITIONDEFFILE
BEGINARCHIVE	GETDBSTATE	PUTALLREPLCELLS
BEGININCBUILDDIM	GETDBSTATS	PUTUPDATEDREPLCELLS
BUILDDIM	GETDEFAULTCALC	REMOVELOCKS
CALC	GETMBCALC	REMOVEUSER
CALCDEFAULT	GETMBRINFO	RENAMEAPP
CALCLINE	GETMEMBERS	RENAMEDB
COPYAPP	GETPARTITIONOTLCHANGES	RENAMEFILTER
COPYDB	GETPARTITIONOTLCHANGESEX	RENAMEOBJECT
COPYFILTER	GETPERFSTATS	RENAMEUSER
COPYOBJECT	GETUPDATEDREPLCELLS	REPORT
CREATEAPP	GETUSERINFO	REPORTLINE
CREATEDB	GETVERSION	RESETDB
CREATEGROUP	GOTO	RESETOTLCHANGETIME
CREATELOCATION	IFERROR	RESETPERFSTATS
CREATEUSER	IMPORT	RESETSTATUS
CREATEVARIABLE	INCBUILDDIM	RUNCALC
DELETEAPP	LISTALIASES	RUNREPT
DELETEDB	LISTAPP	SELECT
DELETEGROUP	LISTDB	SETALIAS
DELETELOCATION	LISTFILES	SETAPPSTATE
DELETELOG	LISTFILTERS	SETDBSTATE
DELETEUSER	LISTGROUPS	SETDBSTATEITEM
DELETEVARIABLE	LISTGROUPUSERS	SETDEFAULTCALC
DISABLELOGIN	LISTLINKEDOBJECTS	SETDEFAULTCALCFILE
DISPLAYALIAS	LISTLOCATIONS	SETLOGIN
ENABLELOGIN	LISTLOCKS	SETMSGLEVEL
ENDARCHIVE	LISTLOGINS	SETPASSWORD
ENDINCBUILDDIM	LISTOBJECTS	SHUTDOWNSERVER
ESTIMATEFULLDBSIZE	LISTUSERS	SLEEP

ADDUSER	GETCRTYPE	PRINTPARTITIONDEFFILE
EXIT	LISTVARIABLES	UNLOADALIAS
EXPORT	LOADALIAS	UNLOADAPP
GETALLREPLCELLS	LOADAPP	UNLOADDB
GETAPPACTIVE	LOADDATA	UNLOCKOBJECT
GETAPPINFO	LOADDB	UPDATE
GETAPPSTATE	LOGIN	UPDATEBAKFILE
GETATTRIBUTESPCS	LOGOUT	UPDATEFILE
GETATTRINFO	LOGOUTALLUSERS	UPDATEVARIABLE
GETCRDB	LOGOUTUSER	VALIDATE
GETCRDBINFO	OUTPUT	VALIDATEPARTITIONDEFFILE
GETCRRATE	PAREXPORT	

ADDUSER

Adds a user to a group.

Syntax

```
ADDUSER groupName userName
```

Parameter	Description
-----------	-------------

<i>groupName</i>	Name of a group.
------------------	------------------

<i>userName</i>	Name of a user.
-----------------	-----------------

Example

To add TomT to MARKETING:

```
ADDUSER "Marketing" "TomT";
```

APPLYOTLCHANGEFILE

Applies the source outline changes specified in the .CHG log file to the target database's outline you selected with the SELECT command.

If the database has multiple partitions of the same type to the same target database or from the same source database, use [APPLYOTLCHANGEFILEEX](#) instead, and specify the data direction.

Syntax

```
APPLYOTLCHANGEFILE numFiles fileName
```


Parameter Description

numFiles	A numeric value indicating the number of .CHG log files to read.
Filename	The name of the .CHG log file to read. The filename must be the full path name of the desired change file on the target database. The filename must be in quotation marks (see example below). More than one file can be specified.

Notes

When the source database outline is modified, the GETPARTITIONOTLCHANGES command records the outline changes to a .CHG file in the source database directory. Therefore, use APPLYOTLCHANGEFILE after calling GETPARTITIONOTLCHANGES. Specify the full path to the source database's .CHG file.

Example

Samppart Company, the target database, is selected. Apply outline changes from Sampeast East, the source database.

```
APPLYOTLCHANGEFILE "1" "C:\Hyperion\products\Essbase\EssbaseServer\app\Sampeast\East\ess00004.chg";
```

See Also

- [GETPARTITIONOTLCHANGES](#)
- [APPLYOTLCHANGEFILEEX](#)

APPLYOTLCHANGEFILEEX

Applies the source outline changes specified in the .CHG log file to the target database's outline you selected with the SELECT command.

Syntax

```
APPLYOTLCHANGEFILEEX numFiles fileName dataFlowDirection
```

Parameter	Description
numFiles	A numeric value indicating the number of .CHG log files to read.
Filename	The name of the .CHG log file to read. The filename must be the full path name of the desired change file on the target database. The filename must be in quotation marks (see example below). More than one file can be specified.
dataFlowDirection	The half of the partition to which you are currently connected: 1 - Source 2 - Target

Notes

When the source database outline is modified, the GETPARTITIONOTLCHANGES command records the outline changes to a .CHG file in the source database directory.

Use APPLYOTLCHANGEFILEEX after calling GETPARTITIONOTLCHANGES. Specify the full path to the source database's .CHG file.

Example

Samppart Company, the target database, is selected. Apply outline changes from Sampeast East, the source database.

```
APPLYOTLCHANGEFILE "1"  
"C:\Hyperion\products\Essbase\EssbaseServer\app\Sampeast\East\ess00004.chg" "1";
```

See Also

- [GETPARTITIONOTLCHANGES](#)

BEGINARCHIVE

Places a database in read-only mode for archiving.

Syntax

```
BEGINARCHIVE App DB file
```

Parameter Description

App	Name of the application.
DB	Name of the database.
file	File to contain the archive.

Notes

Changing the server mode to Read-only allows the database administrator to use an archiving program to back up files on the server. This also prevents writing to files during backup.

The server's Read-only state persists until it is changed back to Read-write with the ENDARCHIVE command. Unless you reset the Read-only state, it persists even after termination of the current session.

The database files to back up are listed in the application\database directory specified by the filename parameter. The archived data overwrites the information in the specified file, if the file already exists. See the *Oracle Essbase Database Administrator's Guide* for more information about restructuring and backup files.

Example

```
BEGINARCHIVE "Sample" "Sales" "June";
```

See Also

- [ENDARCHIVE](#)

BEGININCBUILDDIM

Prepares Essbase Services for deferred-restructure dimension building commands.

Syntax

```
BEGININCBUILDDIM
```

Notes

Deferred-restructure dimension builds have also been called incremental dimension builds. This command works in conjunction with the ENDINCBUILDDIM command to group together one or more INCBUILDDIM statements.

This command locks the outline file. If the outline file is already locked, this command returns an error.

This command copies the outline file (.OTL) to a backup file name (.OTN). Subsequent INCBUILDDIM commands operate on the .OTN file. See the *Oracle Essbase Database Administrator's Guide* for more information about restructuring and backup files.

BEGININCBUILDDIM starts a programming block; ENDINCBUILDDIM ends the programming block.

Example

To build the dimensions specified in GENREF.RUL and LEVELMUL.RUL, discard all data, and save the new outline after the dimension builds are complete:

```
BEGININCBUILDDIM;  
INCBUILDDIM 2 "GENREF.RUL" 2 "GENREF.TXT" 4 "ERR.OUT" 1;  
INCBUILDDIM 2 "LEVELMUL.RUL" 2 "LEVELMUL.TXT" 4 "ERR.OUT" 1;  
ENDINCBUILDDIM 4;
```

See Also

- [ENDINCBUILDDIM](#)
- [INCBUILDDIM](#)
- [BUILDDIM](#)

BUILDDIM

Dynamically builds one or more dimensions from a data file or SQL source.

Syntax

```
BUILDDIM location rulobjName dataLoc sourceName fileType errorLog
```

Parameter	Description
-----------	-------------

location	Location of the rules file. 1 - Local/client. 2 - Remote/server. 3 - File. Use the file is not an Essbase artifact, or if you want to specify the full path name. Otherwise, Essbase looks in the <APPNAME>/<DBNAME> directory.
----------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Parameter	Description
ruleObjName	Name of the rules file
dataLoc	Location of the data file. 1 - Local/client 2 - Remote/server 3 - File. Use if the file is not an Essbase artifact, or if you want to specify the full path name. Otherwise, Essbase looks in the <APPNAME>/<DBNAME> directory. 4 - SQL source.
sourceName	Source of the data file. If <i>dataLoc</i> is 1, 2, or 3, specify the data file name. If <i>dataLoc</i> is 4, specify the SQL user name and password.
fileType	Data file type. 1 - Excel 2 - Lotus .WK1 file (No longer supported) 3 - Lotus .WK3 file (No longer supported) 4 - Text. 5 - Lotus .WK4 file (No longer supported) This parameter is not required if you are using an SQL source.
errorLog	Name of the text file to receive error messages and rejected records.

Notes

This command builds one or more dimensions from a data file or an SQL source. Many applications have large dimensions that are impractical to manually define and maintain. This command makes it possible to automate the dimension-building and updating processes. See the INCBUILDDIM command for another way to build dimensions.

The INCBUILDDIM command is identical to the BUILDDIM command, except for the following:

- INCBUILDDIM does not automatically restructure the database after modifying the dimensions. You can have several consecutive INCBUILDDIM, commands inside a BEGININCBUILDDIM...ENDINCBUILDDIM block. Essbase restructures when it encounters ENDINCBUILDDIM.
- INCBUILDDIM lets you append to, rather than overwrite, the error log.

Example

To build the dimensions as defined by the rules file, PROD.RUL:

```
BUILDDIM 1 "PROD" 1 "PRODUCTS" 4 "PRODERR";
```

To build the dimensions from an SQL table defined in the rules file, PROD.RUL:

```
BUILDDIM 1 "PROD" 4 "TomT" "Password" "PRODERR";
```

CALC

Executes one or more calculation strings.

Syntax

```
CALC "calcString; [calcString;] "
```

Parameter Description

calcString A calculation string (any valid string that is accepted by a calculation script).

Notes

In a batch file, if you include multiple calculation strings in one CALC command, place all of the calculation string parameters in one set of quotation marks and end each command string with a semicolon statement terminator (;). All text within the quotation marks is passed to the calculator.

As an alternate to including multiple calculation strings in this command, place the strings in a calculation script, then call RUNCALC to run the script.

Example

To issue the CALC ALL command:

```
CALC "Calc All;";
```

To calculate the members January and Product:

```
CALC "Jan; Product;";
```

See Also

- [CALCLINE](#)

CALCDEFAULT

Calculates using the default database calculation.

Syntax

```
CALCDEFAULT
```

Notes

This command calculates the relationships defined in the outline, or executes the default calculation.

Example

```
CALCDEFAULT;
```

See Also

- [CALC](#)
- [CALCLINE](#)
- [SETDEFAULTCALC](#)

CALCLINE

Executes a single calculation string.

Syntax

```
CALCLINE calcString
```

Parameter Description

calcString A calculation string (any valid string that is accepted by a calculation script).

Notes

This command executes a single calculation string. In a batch file, place the calculation string parameter in quotation marks and end the string with a semicolon statement terminator (;). All text within the quotation marks is passed to the calculator. This command requires quotation marks.

Example

To issue the CALC ALL command:

```
CALCLINE "Calc All;";
```

To calculate the members January and Product:

```
CALCLINE "Jan; Product;";
```

See Also

- [CALC](#)

COPYAPP

Copies an application.

Syntax

```
COPYAPP sourceApp destApp
```

Parameter Description

sourceApp Name of application to copy.

destApp Name of new application.

Example

```
COPYAPP "FINANC95" "FINANC96";
```

COPYDB

Copies a database.

Syntax

```
COPYDB sourceApp sourceDb destApp destDb
```

Parameter Description

sourceApp Name of the application for the database to copy.

sourceDb Name of the database to copy.

destApp Name of the application for the new database.

destDb Name of the new database.

Example

```
COPYDB "FINANC95" "SALES95" "FINANC96" "SALES96";
```

COPYFILTER

Copies a filter.

Syntax

```
COPYFILTER sourceApp sourceDb sourceFilter destApp destDb destFilter
```

Parameter Description

sourceApp Name of the application that includes the filter to copy.

sourceDb Name of the database that includes the filter to copy.

sourceFilter Name of the filter to copy.

destApp Name of the application for the new filter.

destDb Name for the database for the new filter.

destFilter Name of the filter copy.

Example

```
COPYFILTER "FINANC95" "SALES95" "FILTER95" "FINANC96" "SALES96" "FILTER96";
```

COPYOBJECT

Copies a database artifact.

Syntax

```
COPYOBJECT objType sourceApp sourceDb sourceObj destApp destDb destObj
```

Parameter Description

objType	Type of artifact to list. 0 - Abort 1 - Outline object (not available) 2 - Calculation script 3 - Report script 4 - Rules file 5 - Alias table 6 - Structure file 7 - Backup file (not available) 8 - Worksheet of any type (not available) 9 - Text object 10 - Partition 11 - Linked Reporting Object 12 - Selection 13 - Wizard
sourceApp	Name of the application that includes the artifact to copy.
sourceDb	Name of the database that includes the artifact to copy.
sourceObj	Name of the artifact to copy.
destApp	Name of the application for the new artifact.
destDb	Name of the database for the new artifact.
destObj	Name of the artifact copy.

Notes

objType parameter values 6 and 7 are deprecated.

Example

```
COPYOBJECT "2" "FINANC95" "SALES95" "OLDOBJ" "FINANC96" "SALES96" "NEWOBJ";
```

CREATEAPP

Creates a new application.

Syntax

```
CREATEAPP appName
```

Parameter Description

appName	Name of the application.
---------	--------------------------

Example

To create an application called TBC:

```
CREATEAPP "TBC";
```


CREATEDB

Creates a database.

Syntax

```
CREATEDB appName dbName
```

Parameter Description

appName Name of the application in which to create a database.

dbName Name of the database.

Example

To create an database called FINANCE under an application named TBC:

```
CREATEDB "TBC" "FINANCE";
```

CREATEGROUP

Creates a group.

Syntax

```
CREATEGROUP groupName
```

Parameter Description

groupName Name of the group to create.

Notes

This command creates a new group.

Example

To create a group called MARKETING:

```
CREATEGROUP "MARKETING";
```

CREATELOCATION

Creates a new location alias.

Location aliases provide a shorthand way of managing login information for Essbase databases. Location aliases are mapped to a host name, application name, database name, user name, and password.

Syntax

```
CREATELOCATION alias host application database user_name password
```

Parameter Description

alias	Location alias name.
host	Host name.
application	Application name.
database	Database name.
user_name	Login name.
password	Password for <i>user_name</i> .

Notes

- You can use location aliases only with the @XREF function.
- You must have Database Manager permission to create location aliases.

Example

```
CREATELOCATION "ALIAS3" "LOCAL" "SAMPLE" "BASIC" "TomT" "PASSWORD" ;
```

See Also

- [DELETELOCATION](#)
- [LISTLOCATIONS](#)

CREATEUSER

Creates a new Essbase user ID.

Syntax

```
CREATEUSER userName password
```

Parameter Description

userName	Name of the user.
password	Password for the new user. If the string contains blanks, it must be enclosed in double quotation marks. Leading or trailing spaces are illegal and will be trimmed off. Do not enclose the password in single quotation marks unless you want them to be part of the password.

Example

To create a user named DANTE with the password INFERNO:

```
CREATEUSER "DANTE" "INFERNO" ;
```

CREATEVARIABLE

Defines a substitution variable and its corresponding string value.

Syntax

```
CREATEVARIABLE variableName serverName [appName [dbName]] value
```

Parameter	Description
<i>variableName</i>	Name of the substitution variable. Must be alphanumeric and can contain a maximum of 80 characters. You can use underscores, but not spaces.
<i>serverName</i>	Host name of the Essbase Server.
<i>appName</i>	Optional. Name of the application. If omitted, empty quotes must be used in a script to take its place. (" ")
<i>dbName</i>	Optional. Name of the database. If omitted, empty quotes must be used in a script to take its place. (" ")
<i>value</i>	The string value for the variable. Must be alphanumeric and can contain a maximum of 255 characters. It can include a null value. Do not use the & character as the leading character.

Notes

If you specify only the Essbase Server host name, the variable applies to all applications and databases on the Essbase Server. If you specify the Essbase Server host name and the application name, the variable applies to all databases within the specified application. If you specify the Essbase Server host name, application name, and database name, the variable is for the specified database.

Before you create a new variable, check the names of existing variables with the LISTVARIABLES command. It is possible to overwrite the string value of an existing variable if you create a variable with the same name as the existing variable.

Example

The following command in an ESSCMD script creates a substitution variable on the Sample Basic database, on a host computer named Bamboo. The variable is named CurQtr and has a value of Qtr1.

```
CREATEVARIABLE "CurQtr" "Bamboo" "Sample" "Basic" "Qtr1";
```

The following ESSCMD script creates a substitution variable that applies to all applications and databases on the Essbase Server named Aspen. Application and database input is left blank because the variable is system-wide; however, the empty quotation marks are still required as placeholders.

```
login "Aspen" "fiona" "sunflower";  
CREATEVARIABLE "CurQtr" "aspen" "" "" "Qtr4";
```

See Also

- [LISTVARIABLES](#)
- [UPDATEVARIABLE](#)

DELETEAPP

Deletes an application.

Syntax

```
DELETEAPP appName
```

Parameter Description

appName Name of the application to delete.

Notes

Deleting an application deletes all of its associated databases and other artifacts, along with any additional files that reside in the application and database directories.

Example

To delete an application called TBD:

```
DELETEAPP "TBC" ;
```

DELETEDB

Deletes a database.

Syntax

```
DELETEDB appName dbName
```

Parameter Description

appName Name of the application containing the database to delete.

dbName Name of the database.

Notes

Deleting a database deletes all of its associated artifacts, along with any additional files that reside in the database directory.

Example

To delete a database called BASIC from an application called TBC:

```
DELETEDB "TBC" "BASIC" ;
```

DELETEGROUP

Deletes a group.

Syntax

```
DELETEGROUP groupName
```

Parameter	Description
-----------	-------------

groupName	Name of the group to delete.
-----------	------------------------------

Notes

This command deletes an Essbase security group. Deleting the group does not delete users that were in the group.

Example

To delete a group called MARKETING:

```
DELETEDGROUP "MARKETING";
```

See Also

- [DELETEUSER](#)
- [REMOVEUSER](#)

DELETELOCATION

Removes a location alias from the current database.

Syntax

```
DELETELOCATION alias
```

Parameter	Description
-----------	-------------

alias	Name of location alias.
-------	-------------------------

Notes

You must have Database Manager privilege to delete location aliases.

Example

```
DELETELOCATION "ALIAS3";
```

See Also

- [CREATELOCATION](#)
- [LISTLOCATIONS](#)

DELETELOG

Deletes accumulated entries from an application log file (*appName/log*) or the Essbase Server log file (*essbase.log*).

Syntax

```
DELETELOG appName
```

Parameter Description

`appName` Name of application. If you omit *appName*, Essbase clears the Essbase Server log file.

Notes

Each application has a log file, which records all user requests and activities in all databases in the application. The log file should be cleared regularly to prevent its becoming too large.

In addition, there is an Essbase Server log file, which records all the commands displayed in the main Essbase Server Agent window.

Example

To clear the log file of an application called SAMPLE:

```
DELETELOG "SAMPLE" ;
```

To clear the Essbase Server log file:

```
DELETELOG " "
```

DELETEUSER

Deletes an Essbase user ID.

Syntax

```
DELETEUSER userName
```

Parameter Description

`userName` Name of the user to delete.

Notes

- Deleting the user ID deletes the user from the list of users on the Essbase Server, as well as logging the user out of the active session.
- If you want to remove a user from a group without removing the user, use [REMOVEUSER](#) instead.
- Do not include a group name in the DELETEUSER command line; otherwise, the group will also be deleted.

Example

To delete a user named DANTE:

```
DELETEUSER "DANTE" ;
```

See Also

- [DELETEGROUP](#)
- [REMOVEUSER](#)

DELETEVARIABLE

Removes a substitution variable.

Syntax

```
DELETEVARIABLE variableName serverName [appName [dbName]]
```

Parameter	Description
-----------	-------------

<i>variableName</i>	Name of substitution variable to delete.
---------------------	------------------------------------------

<i>serverName</i>	Name of the server.
-------------------	---------------------

<i>appName</i>	Optional. Name of the application.
----------------	------------------------------------

<i>dbName</i>	Optional. Name of the database.
---------------	---------------------------------

Notes

If the variable was created at the server level, specify only the server name. If the variable was created at the application level, specify the server and application. If the variable was created at the database level, select the server, application, and database.

Example

```
DELETEVARIABLE "CurQtr" "Bamboo" "Sample" "Basic";
```

DISABLELOGIN

Prevents users from logging in to databases in an application. Administrators and application managers for the application are not affected by this setting, but other connected users are affected.

Syntax

```
DISABLELOGIN [appName]
```

Parameter	Description
-----------	-------------

<i>appName</i>	Optional. Required only if no application is selected.
----------------	--------------------------------------------------------

Notes

Issue the DISABLELOGIN command to prevent users from accessing databases in an application during maintenance. Administrators and application managers are not affected.

The DISABLELOGIN command prevents any user with a permission lower than Application Manager from making connections to the databases that require the databases to be started. This includes starting the databases or performing the [SELECT](#) command on the databases.

Database connections remain disabled until re-enabled by as follows:

- By the administrator, using [ENABLELOGIN](#).

- By the administrator, using application settings. In Administration Services, select the Allow Connects check box under the Security node in the Application Properties window - General tab.

By default, connections are enabled.

Example

```
DISABLELOGIN;
```

See Also

- [ENABLELOGIN](#)

DISPLAYALIAS

Lists the alias names defined in an alias table.

Syntax

```
DISPLAYALIAS aliasTableName
```

Parameter	Description
-----------	-------------

<i>aliasTableName</i>	Name of the alias table.
-----------------------	--------------------------

Example

To display the alias names defined in an alias table called DEFAULT:

```
DISPLAYALIAS "DEFAULT" ;
```

ENABLELOGIN

Enables connections to databases in an application.

Syntax

```
ENABLELOGIN [appName]
```

Parameter	Description
-----------	-------------

<i>appName</i>	Optional. Required only if no application selected.
----------------	-----------------------------------------------------

Notes

This command reverses the effect of [DISABLELOGIN](#).

Example

```
ENABLELOGIN;
```


ENDARCHIVE

Restores the database to read-write mode after archiving is complete.

Syntax

```
ENDARCHIVE appName dbName
```

Parameter Description

appName Name of the application containing the archived database.

dbName Name of the database.

Notes

After you call [BEGINARCHIVE](#), use ENDARCHIVE to restore the database to read-write mode. Otherwise, the read-only state persists even after the termination of the current session. See the *Oracle Essbase Database Administrator's Guide* for more information about restructuring and backup files.

Example

```
ENDARCHIVE;
```

See Also

- [BEGINARCHIVE](#)

ENDINCBUILDDIM

Ends the programming block started by BEGININCBUILDDIM and restructures the database after one or more deferred-restructure dimension-building (INCBUILDDIM) commands. Deferred restructure dimension builds have also been called incremental dimension builds.

Syntax

```
ENDINCBUILDDIM preserve
```

Parameter Description

preserve Specifies whether to preserve existing data in the database. This parameter is required. Values:

- 1 - Preserves all existing data blocks.
- 2 - Preserves existing level 0 data.
- 3 - Preserves existing input-level data.
- 4 - Discards all existing data.

Notes

This command works in conjunction with the BEGININCBUILDDIM command to group together one or more INCBUILDDIM statements.

This command restructures the database according to the dimension changes that occur as a result of the INCBUILDDIM commands.

This command preserves existing data according to the preserve option.

This command unlocks the outline once restructuring is complete, and overwrites the original .OTL file with the newly modified .OTN file. See [BEGININCBUILDDIM](#) for information.

If one or more of the INCBUILDDIM commands that precede the ENDINCBUILDDIM command fails, ENDINCBUILDDIM still restructures the database.

WARNING: If you don't issue an ENDINCBUILDDIM command after a BEGININCBUILDDIM command and one or more INCBUILDDIM commands, the changes made to the .OTN file are not copied to the database outline (.OTL) file, and the data is not restructured.

Example

To build the dimensions specified in GENREF.RUL and LEVELMUL.RUL, discard all data, and save the new outline after the dimension builds are complete:

```
BEGININCBUILDDIM;  
  INCBUILDDIM 2 "GENREF.RUL" 2 "GENREF.TXT" 4 "ERR.OUT" 1;  
  INCBUILDDIM 2 "LEVELMUL.RUL" 2 "LEVELMUL.TXT" 4 "ERR.OUT" 1;  
ENDINCBUILDDIM 4;
```

See Also

- [BEGININCBUILDDIM](#)
- [INCBUILDDIM](#)

ESTIMATEFULLDBSIZE

Estimates the number of blocks a full calculation ([CALC ALL](#)) of the database creates, based on the number of blocks that exist before calculation. The database can have all data loaded, or a random sampling of data.

Syntax

```
ESTIMATEFULLDBSIZE
```

Notes

- Use this estimate to help you plan disk space requirements.
- Outlines that contain sparse formulas or topdown formulas are not supported.
- Select an application and database before issuing this command.

Example

Assume that you have fully loaded Sample Basic. Use this command before calculation to predict the number of blocks that would be created.

```
estimatefulldbsize;
```

```
Estimated count of blocks after full calculation = 335
```

```
Time elapsed to calculate this estimation = 0.02 seconds
```

EXIT

Terminates the current session of the ESSCMD utility.

Syntax

```
EXIT
```

Example

```
EXIT;
```

EXPORT

Writes the data values of a database to a text file.

Syntax

```
EXPORT exportName amount formatOption
```

Parameter	Description
-----------	-------------

<code>exportName</code>	Specifies the name, including the path, of the file for the exported data. If no path is specified, the file is created in the <code>ARBORPATH\app</code> directory.
-------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------

<code>amount</code>	Specifies the number representing the data to export.
---------------------	-------------------------------------------------------

- 1 - All data
- 2 - Only level 0 blocks
- 3 - Only data from blocks with input data

<code>formatOption</code>	Specifies the format of the data.
---------------------------	-----------------------------------

- 0 (null) - Non-columnar format. This is the default.
- 1 - Columnar format

Notes

The EXPORT command copies data in text format as an alternative to database archiving wherein you copy the files in binary format. Text format is more easily ported to other databases, and users can easily read it. This command exports only the data, rather than the entire database. Because only data is exported, it is more complex to restore the database from an export file than from a true database archive. You must reload and recalculate the data if you use an export file to restore a database.

Some file systems do not support text files larger than 2 GB. If the exported data exceeds 2 GB, Essbase creates multiple export files, as needed. An underscore and number is appended to the file names of the additional files, starting with `_1`. For example, if `exportName` is `outfile.txt` and three files are created, the resulting file names are `outfile.txt`, `outfile_1.txt`, and `outfile_2.txt`.

The export process does not begin until all users are logged out of the database. After the export process begins, the database is in read-only mode. After the export process is finished, Essbase returns the database to read-write mode.

The EXPORT command works on both aggregate storage and block storage databases; however, aggregate storage exports work differently from block storage exports. See *Oracle Essbase Database Administrator's Guide*.

Example

To create an export file called E060693 that contains only level zero data in columnar format:

```
EXPORT "E060693" 2 1;
```

If the exported data in this example exceeds 4 GB, three files are created: E060693, E060693_1, and E060693_2.

See Also

- [PAREXPORT](#)
- [DATAEXPORT](#)

GETALLREPLCELLS

The GETALLREPLCELLS command replicates all data cells in the replicated partition from a source database to a target database. Use this command when you are in the data target database.

Syntax

1:

```
GETALLREPLCELLS sourceServerName sourceAppName sourceDbName
```

2:

```
GETALLREPLCELLS ALL
```

Parameter	Description
-----------	-------------

sourceServer	Host name of the Essbase Server data source.
--------------	----------------------------------------------

sourceApp	Name of the data source application.
-----------	--------------------------------------

sourceDb	Name of the data source database.
----------	-----------------------------------

ALL	Updates cells for all partitions where the selected database is a data replication target.
-----	--------------------------------------------------------------------------------------------

Notes

This command gets all replicated data cells from the Essbase Server data source, application, and database, and replicates them in the data target database you select with the SELECT command. This is useful when the data source and data target databases need to be resynchronized.

GETALLREPLCELLS gets cells from the data source to the data target, based on a request made from the data target; PUTALLREPLCELLS puts cells from the data source to the data target, based on a request made from the data source.

Example

```
GETALLREPLCELLS "Aspen" "Sample" "Basic";
```

See Also

- [GETUPDATEDREPLCELLS](#)
- [PUTALLREPLCELLS](#)

GETAPPACTIVE

Returns the name of the currently selected application.

Syntax

```
GETAPPACTIVE
```

Example

The following example shows the command and its results:

```
GETAPPACTIVE;  
GetAppActive:
```

```
Current active application is [sample]
```

See Also

- [GETAPPINFO](#)
- [GETAPPSTATE](#)

GETAPPINFO

Returns host, user, and database information for the current application.

Syntax

```
GETAPPINFO [appName]
```

Parameter Description

appName Optional. Required only if no application is selected.

Notes

This command returns the following information about the application: name, Essbase Server host name, status, elapsed time, users connected, number of databases, and a list of all databases.

Example

```
GETAPPINFO;
```

See Also

- [GETAPPACTIVE](#)
- [GETAPPSTATE](#)

GETAPPSTATE

Returns information on the state of the currently selected application.

Syntax

```
GETAPPSTATE [appName]
```

Parameter Description

appName Optional. Required only if no application is selected.

Notes

This command returns information on the state of the currently selected application, as follows:

Loadable (Y/N) AutoLoad (Y/N), Access Level, Allow Connects (Y/N), Allow Commands (Y/N), Allow Updates (Y/N), Security, Lock Timeout, LRO File Size Limit.

Example

```
GETAPPSTATE;
```

See Also

- [GETAPPACTIVE](#)
- [GETAPPINFO](#)

GETATTRIBUTE SPECS

Returns the current attribute specifications for the selected application and database.

Syntax

```
GETATTRIBUTE SPECS
```

Notes

This command returns the current attribute specifications for the application and database, including attribute member name format, Attribute Calculation dimension member names, Boolean and date member names, and numeric range specifications.

Example

```
GETATTRIBUTE SPECS;
```

Returns:

```
-----Attribute Specifications-----
```

```
Prefix/Suffix      : Prefix
Use Name of        : Parent
Delimiter          : '_'
Date Format         : MM-DD-YYYY
Bucketing Type     : Upper Bound inclusive
Default for TRUE   : True
Default for FALSE  : False
```

```

Default for Attr Calc : Attribute Calculations
Default for Sum       : Sum
Default for Count     : Count
Default for Average   : Average
Default for Min       : Min
Default for Max       : Max

```

The name of the attribute level 0 member 3000000, in the following dimension structure, varies depending on the attribute member name format.

Attribute Member:

```

Population
  Small
    3000000

```

Sample Name Variations:

Settings	Resulting Member name
Prefix/Suffix : Prefix Use Name of : None Delimiter : '_'	3000000
Prefix/Suffix : Prefix Use Name of : Parent Delimiter : '_'	Small_3000000
Prefix/Suffix : Prefix Use Name of : All Ancestors Delimiter : '^'	Population^Small^3000000

GETATTRINFO

Returns member, dimension, and name information for given attribute members:

- MbrName: Member name.
- DimName: Dimension of which the attribute is a member.
- Data Type: The attribute member type. Values: Boolean, date, numeric, text.
- Data Value: The short attribute member name, if an attribute member. This is the name shown in the Outline Editor. For example, Data Value = 20 for the attribute member named Ounces_20.

Syntax

```
GETATTRINFO mbrName
```

Parameter Description

mbrName Full attribute member name. Example:

```
Intro Date_07-26-1996
```

Notes

- The output of this command is a subset of the output for the GETMBRINFO command.
- To learn the exact format of the attribute member name, you can enter [GETMEMBERS](#) *<parent>*, where *<parent>* is the parent of the attribute member. For example, `GETMEMBERS "Intro Date";` returns:

```
Intro Date_12-10-1996  Intro Date_10-01-1996  Intro Date_07-26-1996
Intro Date_06-26-1996  Intro Date_04-01-1996  Intro Date_03-25-1996
Intro Date_09-27-1995  Intro Date
```

Example

```
GETATTRINFO "Caffeinated_True";
```

Returns:

```
Member info of [caffeinated_true]
-----
MbrName      : Caffeinated_True
DimName      : Caffeinated
Attribute Type : Boolean
Attribute Value : True
```

GETCRDB

Returns the name of the currency database linked to the currently selected database.

Syntax

```
GETCRDB
```

Example

```
GETCRDB;
```

See Also

- [GETCRDBINFO](#)
- [GETCRRATE](#)
- [GETCRTYPE](#)

GETCRDBINFO

Returns information about the currency database linked to the currently selected database.

Syntax

```
GETCRDBINFO
```


Example

```
GETCRDBINFO;
```

GETCRRATE

Returns the currency rate for currency partitions.

Syntax

```
GETCRRATE
```

Example

```
GETCRRATE;
```

See Also

- [GETCRDB](#)
- [GETCRDBINFO](#)

GETCRTYPE

Returns information about the default currency type and conversion method.

Syntax

```
GETCRTYPE
```

Example

```
GETCRTYPE;
```

See Also

- [GETDBINFO](#)
- [GETDBSTATS](#)

GETDBACTIVE

Returns the name of the currently selected database.

Syntax

```
GETDBACTIVE
```

Example

```
GETDBACTIVE;
```

GETDBINFO

Returns information on the state of the currently selected database.

Some settings do not take effect until the database is re-started. For information on most recently entered settings, see [GETDBSTATE](#).

Syntax

```
GETDBINFO [appName dbName]
```

Parameter	Description
-----------	-------------

appName dbName Optional. Both required if no application and database are selected.

Notes

When working with currency databases, values viewed using [GETDBSTATE](#) and [GETDBINFO](#) may differ from each other. The currency database may temporarily inherit attributes from its associated database. To have the values match, issue the appropriate [SETDBSTATEITEM](#) command.

Example

```
GETDBINFO;
```

Returns:

```
----- Database Information -----
Name                               : Basic
Application Name                   : Sample
Database Type                      : NORMAL
Status                             : Loaded
Elapsed Db Time                   : 00:01:38:31
Users Connected                   : 2
Blocks Locked                     : 0
Dimensions                        : 10
Data Status                       : Data has been modified
                                   since last calculation.
Data File Cache Size Setting      : 33554432
Current Data File Cache Size     : 8388608
Data Cache Size Setting          : 3144960
Current Data Cache Size          : 2096064
Index Cache Size Setting         : 10485760
Current Index Cache Size         : 10485760
Index Page Size Setting          : 1024
Current Index Page Size          : 8192
Cache Memory Locking             : Enabled
Database State                   : Read-write
Data Compression on Disk         : Yes
Data Compression Type            : BitMap Compression
Retrieval Buffer Size (in K)      : 10
Retrieval Sort Buffer Size (in K) : 10
Isolation Level                  : Uncommitted Access
Pre Image Access                 : No
Time Out                         : Never
Number of blocks modified before internal commit : 3000
Number of rows to data load before internal commit : 0
Number of disk volume definitions : 0
```

```
--Currency Info--
```

```
Currency Country Dimension Member      :
Currency Time Dimension Member         : Year
Currency Category Dimension Member     : Measures
Currency Type Dimension Member         :
Currency Partition Member              :
```

--Request Info--

```
Request Type      : Data Load
User Name        : admin
Start Time       : Mon Feb 17 11:42:59 2004
End Time         : Mon Feb 17 11:43:22 2004
Request Type     : Default Calculation
User Name        : admin
Start Time       : Mon Feb 17 12:57:45 2004
End Time         : Mon Feb 17 12:57:46 2004
Request Type     : Outline Update
User Name        : admin
Start Time       : Wed Jan 22 12:39:27 2004
End Time         : Wed Jan 22 12:39:30 2004
```

See Also

- [GETDBSTATE](#)

GETDBSTATE

Returns the most recently entered database settings for the selected database.

For settings currently in effect, see [GETDBINFO](#).

Syntax

```
GETDBSTATE [appName dbName]
```

Parameter	Description
-----------	-------------

<i>appName dbName</i>	Optional. Both required if no application and database are selected.
-----------------------	----------------------------------------------------------------------

Notes

When working with currency databases, values viewed using GETDBSTATE and GETDBINFO may differ from each other. The currency database may temporarily inherit attributes from its associated database. To have the values match, issue the appropriate [SETDBSTATEITEM](#) command.

Example

```
GETDBSTATE;
```

Returns:

```
-----Database State-----
```

Description:

```
Allow Database to Start                : Yes
```

```
Start Database when Application Starts : Yes
```

```

Access Level : None
Data File Cache Size : 33554432
Data Cache Size : 3145728
Aggregate Missing Values : No
Perform two pass calc when [CALC ALL;] : Yes
Create blocks on equation : No
Currency DB Name : N/A
Currency Conversion Type Member : N/A
Currency Conversion Type : N/A
Index Cache Size : 1048576
Index Page Size : 8192
Cache Memory Locking : Disabled
Data Compression on Disk : Yes
Data Compression Type : BitMap Compression
Retrieval Buffer Size (in K) : 10
Retrieval Sort Buffer Size (in K) : 10
Isolation Level : Uncommitted Access
Pre Image Access : Yes
Time Out after : 20 sec.
Number of blocks modified before internal commit : 3000
Number of rows to data load before internal commit : 0
Number of disk volume definitions : 0

I/O Access Mode (pending) : Buffered
I/O Access Mode (in use) : Buffered
Direct I/O Type (in use) : N/A

```

See Also

- [GETDBINFO](#)

GETDBSTATS

Returns information about dimensions and data blocks for the selected database.

Syntax

```
GETDBSTATS
```

Notes

For information about how to use the Average Fragmentation Quotient, see the *Oracle Essbase Database Administrator's Guide* section about monitoring performance.

Example

```
GETDBSTATS;
```

Returns:

```

-----Statistics of sample:basic -----

Dimension Name          Type      Declared Size  Actual Size
=====
Year                    DENSE    19             12
Measures                 DENSE    17             8
Product                  SPARSE   22             19
Market                   SPARSE   25             25

```

Scenario	DENSE	5	2
Caffeinated	SPARSE	3	3
Ounces	SPARSE	5	5
Pkg Type	SPARSE	3	3
Population	SPARSE	15	15
Intro Date	SPARSE	8	8

Number of dimensions	:	10
Declared Block Size	:	1615
Actual Block Size	:	192
Declared Maximum Blocks	:	550
Actual Maximum Blocks	:	475
Number of Non Missing Leaf Blocks	:	177
Number of Non Missing Non Leaf Blocks	:	197
Number of Total Blocks	:	374
Index Type	:	B+ TREE
Average Block Density	:	93.75
Average Sparse Density	:	78.73684
Block Compression Ratio	:	0.9552239
Average Clustering Ratio	:	1
Average Fragmentation Quotient	:	0.01238265
Free Space is Recoverable	:	false
Estimated Bytes of Recoverable Free Space	:	0

See Also

- [GETDBACTIVE](#)
- [GETDBINFO](#)
- [GETDBSTATE](#)

GETDEFAULTCALC

Returns the default calculation script of the currently selected database.

Syntax

```
GETDEFAULTCALC
```

Notes

The default calculation script refers to either the relations defined in the database outline (CALC ALL) or to the set of calc strings defined as the default database calculation. This command returns the contents of the calculation script designated as default for the database.

Example

```
GETDEFAULTCALC;
```

Returns:

```
Default Calc Script--
CALC ALL;
```

GETMBRCALC

Returns the calc string associated with the selected member.

Syntax

```
GETMBRCALC mbrName
```

Parameter Description

mbrName Member name

Example

```
GETMBRCALC "Profit %";
```

Returns the following:

```
Outline Defined Calc Equation. [Profit % Sales;]  
Last Calculated Calc Equation. [Profit % Sales;]
```

GETMBRINFO

Returns information on a specific member.

Syntax

```
GETMBRINFO mbrName
```

Parameter Description

mbrName Member name

Notes

This command returns the following information on a specific member:

- Member name.
- Member number.
- Dimension name.
- Dimension number.
- Data-storage share information.
- Level: Steps from bottom to top.
- Generation: Steps from top to bottom.
- Unary operator (+, -, *, /, %, ~) for consolidation: add, subtract, multiply, divide, percentile, ignore.
- Member tag types, if any; for example, Accounts and Time Series tags, Two-Pass Calc tags.
- Name of the tagged currency database member (if any).
- Currency conversion. Values: Yes/No

- Member description.
- Parent member name.
- Child member name.
- Previous member name.
- Next member name.
- Attributed: Whether the member has attributes associated with it. Values: Yes, No, N/A (N/A for attribute members).
- Attribute Type: The attribute member type. Values: Boolean, Date, Numeric, Text.
- Attribute Value: The short attribute member name, if an attribute member. This is the name shown in the Outline Editor.
- Member has relational descendants: Applicable if Hybrid Analysis is used.

Example

```
GETMBRINFO "Ounces_20";
```

Returns:

```
Member info of [Ounces_20]
-----
MbrName       : Ounces_20
MbrNumber     : 2
DimName       : Ounces
DimNumber     : 7
Status        : Virtual Member (Non-stored)
Level         : 0
Generation    : 2
UnaryCalc     : NoRollUp
MbrTagType    : SkipNone
CrMbrName     : N/A
CurrConvert   : N/A
Description   : N/A
ParentMbrName : Ounces
ChildMbrName  : N/A
PrevMbrName   : Ounces_32
NextMbrName   : Ounces_16
Attributed    : N/A
Attribute Type : Numeric(Double)
Attribute Value: 20
Member has relational descendants: No
```

GETMEMBERS

Returns a list of members from the currently selected database.

Syntax

```
GETMEMBERS [mbrString]
```

Parameter Description

mbrString Optional. Dimension or member name. If specified, returns children of named dimension or member. The default is NULL, which returns a list of dimensions in the database.

Example

To return a list of the database dimension names:

```
GETMEMBERS ;
```

To return a list of the children of Product:

```
GETMEMBERS "Product";
```

To return a list of the children of Qtr1:

```
GETMEMBERS "Qtr1";
```

GETPARTITIONOTLCHANGES

Retrieves a list of outline changes made to the partitioned area in the source database, and writes these changes to the .CHG file on the target database you select with the SELECT command.

If the database has multiple partitions of the same type to the same target database or from the same source database, use [GetPartitionOtlChangesEx](#) instead, and specify the data direction.

Note: All arguments must be provided on one line.

Syntax

```
GETPARTITIONOTLCHANGES sourceServerName sourceAppName  
  sourceDbName sourcePartitionType getAllOtlChanges  
 [  
  getAllDimChanges  
  [getNewDim getDeletedDim getUpdatedDim getMovedDim  
  getRenamedDim]  
  getAllMbrChanges  
  [getNewMbrs getDeletedMbrs  
  getRenamedMbrs getMovedMbrs]  
  getAllMbrAttribChanges  
  [getChngedMbrStatus getChngedMbrAlias getChngedMbrCalcSym  
  getChngedMbrAcctType getChngedMbrCurrCnvInfo  
  getChngedMbrUda getChngedMbrCalcFormulas]  
  getChangedLevNbr  
  getChangedGenNbr  
 ]
```

Parameter	Description
sourceServerName	Name of the data source server where the outline changes were made.
sourceAppName	Name of the data source application where the outline changes were made.
sourceDbName	Name of the data source database where the outline changes were made.

Parameter	Description
sourcePartitionType	Name of the partition type where the outline changes were made. Can be any of the following: 1 - Replicated 2 - Linked 3 - Transparent
getAllOtlChanges	Lists all changes to the database outline. Values: Y/N.
getAllDimChanges	Lists all changes to the dimensions, including member names. Values: Y/N.
getNewDim	Lists newly created dimensions. Values: Y/N.
getDeletedDim	Lists deleted dimensions. Values: Y/N.
getUpdatedDim	Lists updated dimensions. Values: Y/N.
getMovedDim	Lists moved dimensions. Values: Y/N.
getRenamedDim	Lists renamed dimensions. Values: Y/N.
getAllMbrChanges	Lists all member changes. Values: Y/N.
getNewMbrs	Lists newly created members. Values: Y/N.
getDeletedMbrs	Lists deleted members. Values: Y/N.
getRenamedMbrs	Lists renamed members. Values: Y/N.
getMovedMbrs	Lists moved members. Values: Y/N.
getAllMbrAttribChanges	Lists all changes to member attributes. Values: Y/N.
getChngedMbrStatus	Lists members that have a changed status such as data storage or Dynamic Time Series information. Values: Y/N.
getChngedMbrAlias	Lists changed member aliases. Values: Y/N.
getChngedMbrCalcSym	Lists changed member unary operators. Values: Y/N.
getChngedMbrAcctType	Lists changed account type information for members in an Accounts dimension. Values: Y/N.
getChngedMbrCurrCnvInfo	Lists changed member currency conversion information. Values: Y/N.
getChngedMbrUda	Lists changed member user-defined attributes. Values: Y/N.
getChngedMbrCalcFormulas	Lists changed member calc formulas. Values: Y/N.
getChangedLevNbr	Lists changed level numbers. Values: Y/N.
getChangedGenNbr	Lists changed generation numbers. Values: Y/N.

Notes

This command retrieves a list of all outline changes made to the data source database, based on the selected parameters, and writes the changes to a .CHG log file on the selected data target

database. Essbase creates the .CHG file, and names it with a file name representing the partition ID.

Example

With Optional Parameters:

```
GETPARTITIONOTLCHANGES "BAMBOO" "SAMPLE" "BASIC"  
"1" "N" "Y" "Y" "Y" "N" "Y" "Y" "N" "Y" "Y" "N" "Y" "N" "Y" "Y" "Y" "Y" "N" "Y" "Y" "Y"  
"Y";
```

Without Optional Parameters:

```
GETPARTITIONOTLCHANGES "BAMBOO" "SAMPLE" "BASIC"  
"1" "N" "Y" "Y" "Y" "Y";
```

See Also

- [APPLYOTLCHANGEFILE](#)
- [GETPARTITIONOTLCHANGESEX](#)

GETPARTITIONOTLCHANGESEX

Retrieves a list of outline changes made to the partitioned area in the source database, and writes these changes to the .CHG file on the target database you select with the SELECT command.

Note: All arguments must be provided on one line.

Syntax

```
GETPARTITIONOTLCHANGESEX sourceServerName sourceAppName  
sourceDbName sourcePartitionType dataFlowDirection getAllOtlChanges  
 [  
  getAllDimChanges  
  [getNewDim getDeletedDim getUpdatedDim getMovedDim  
  getRenamedDim]  
  getAllMbrChanges  
  [getNewMbrs getDeletedMbrs  
  getRenamedMbrs getMovedMbrs]  
  getAllMbrAttribChanges  
  [getChngedMbrStatus getChngedMbrAlias getChngedMbrCalcSym  
  getChngedMbrAcctType getChngedMbrCurrCnvInfo  
  getChngedMbrUda getChngedMbrCalcFormulas]  
  getChangedLevNbr  
  getChangedGenNbr ]
```

Parameter	Description
sourceServerName	Name of the data source server where the outline changes were made.
sourceAppName	Name of the data source application where the outline changes were made.
sourceDbName	Name of the data source database where the outline changes were made.

Parameter	Description
sourcePartitionType	Name of the partition type where the outline changes were made. Can be any of the following: 1 - Replicated 2 - Linked 3 - Transparent
dataFlowDirection	The half of the partition to which you are currently connected: 1 - Source 2 - Target
getAllOtlChanges	Lists all changes to the database outline. Values: Y/N.
getAllDimChanges	Lists all changes to the dimensions, including member names. Values: Y/N.
getNewDim	Lists newly created dimensions. Values: Y/N.
getDeletedDim	Lists deleted dimensions. Values: Y/N.
getUpdatedDim	Lists updated dimensions. Values: Y/N.
getMovedDim	Lists moved dimensions. Values: Y/N.
getRenamedDim	Lists renamed dimensions. Values: Y/N.
getAllMbrChanges	Lists all member changes. Values: Y/N.
getNewMbrs	Lists newly created members. Values: Y/N.
getDeletedMbrs	Lists deleted members. Values: Y/N.
getRenamedMbrs	Lists renamed members. Values: Y/N.
getMovedMbrs	Lists moved members. Values: Y/N.
getAllMbrAttribChanges	Lists all changes to member attributes. Values: Y/N.
getChngedMbrStatus	Lists members that have a changed status such as data storage or Dynamic Time Series information. Values: Y/N.
getChngedMbrAlias	Lists changed member aliases. Values: Y/N.
getChngedMbrCalcSym	Lists changed member unary operators. Values: Y/N.
getChngedMbrAcctType	Lists changed account type information for members in an Accounts dimension. Values: Y/N.
getChngedMbrCurrCnvInfo	Lists changed member currency conversion information. Values: Y/N.
getChngedMbrUda	Lists changed member user-defined attributes. Values: Y/N.
getChngedMbrCalcFormulas	Lists changed member calc formulas. Values: Y/N.
getChangedLevNbr	Lists changed level numbers. Values: Y/N.
getChangedGenNbr	Lists changed generation numbers. Values: Y/N.

Notes

This command retrieves a list of all outline changes made to the data source database, based on the selected parameters, and writes the changes to a .CHG log file on the selected data target database. Essbase creates the .CHG file, and names it with a file name representing the partition ID.

Example

With Optional Parameters:

```
GETPARTITIONOTLCHANGESEX "BAMBOO" "SAMPLE" "BASIC"  
"1" "1" "N" "Y" "Y" "Y" "N" "Y" "Y" "N" "Y" "Y" "N" "Y" "N" "Y" "Y" "Y" "Y" "N" "Y" "Y"  
"Y" "Y";
```

Without Optional Parameters:

```
GETPARTITIONOTLCHANGESEX "BAMBOO" "SAMPLE" "BASIC"  
"1" "1" "N" "Y" "Y" "Y" "Y";
```

See Also

- [APPLYOTLCHANGEFILE](#)

GETPERFSTATS

Returns performance statistics tables.

Syntax

```
GETPERFSTATS
```

Notes

This command returns short, medium, and long performance statistics for the thread, database, and application. The statistics appear as tables in the ESSCMD window. To gather performance statistics, you must first enable statistics gathering using `RESETPERFSTATS`. You also use `RESETPERFSTATS` to return to zero the statistical *persistence* (length) and *scope* (granularity). Collecting and analyzing performance statistics can help you understand whether the databases are in good running condition or could use modifications to improve performance.

For full description of the performance statistics output, see [“Performance Statistics in MaxL” on page 627](#). ESSCMD usage is deprecated.

See Also

- [RESETPERFSTATS](#)

GETUPDATEDREPLCELLS

Replicates all changed data cells in the replicated partition from a data source database to the selected data target database. Use this command when you are in the target database.

Syntax

1:

```
GETUPDATEDREPLCELLS sourceServerName sourceAppName sourceDbName
```

2:

```
GETUPDATEDREPLCELLS ALL
```

Parameter	Description
sourceServerName	Name of the data source server from which cells are replicated.
sourceAppName	Name of the data source application from which cells are replicated.
sourceDbName	Name of the data source database from which cells are replicated.
ALL	Updates cells for all partitions where the selected database is a data replication target.

Notes

This command gets all changed replicated data cells from the data source server, application, and database, and replicates them in the data target database you select with the SELECT command.

Essbase determines what updates are performed, based on an internal time stamp which is read at the block level. Whenever data in the block changes, the time stamp is reset to the current time. If data is changed that is not defined in the replication area, but is part of the data block, the time stamp is still refreshed. Therefore, it is possible to update data in the replication area, even though the replication data has not changed.

When a block is removed by such actions as RESETDB and you request an update of the replication cells, Essbase performs an internal search that identifies blocks without time stamps. Essbase then gets all cells from the replication area, instead of only changed cells, which may cause a time delay.

GETUPDATEDREPLCELLS gets cells from the data source server to the data target server, based on a request made from the data target server; PUTUPDATEDREPLCELLS puts cells from the data source server to the data target server, based on a request made from the data source server.

Example

```
GETUPDATEDREPLCELLS "Aspen" "Sample" "Basic";
```

See Also

- [PUTUPDATEDREPLCELLS](#)
- [GETALLREPLCELLS](#)

GETUSERINFO

Returns information about a specified user or group.

Syntax

```
GETUSERINFO userName
```

Parameter Description

userName Name of the user or group.

Notes

This command returns the following information about a specified user or group:

User/Group name, Logged in (Y/N), Access Level, Last successful login, failed login attempts since then, Login ID.

Example

```
GETUSERINFO "TomT" ;
```

GETVERSION

Returns the version number and patch number information on the current Essbase Server software installation.

Syntax

```
GETVERSION
```

Example

```
GETVERSION;
```

GOTO

Skips all commands until it encounters the associated label.

Syntax

```
GOTO "Label"; <SKIPPED COMMANDS> :Label ; <COMMANDS OR EOF>
```

Parameter Description

"*Label*" A string of characters; not case-sensitive.

:Label Target location, preceded by a colon (:) and associated with "*Label*". Processing skips to this label.

Notes

This command provides unconditional branching. This means that branching occurs regardless of the success or failure of previous commands.

Commands that follow *:Label* can implement error handling or stop processing. Processing skips all subsequent commands and moves to the associated label, where it resumes. Processing ignores even the EXIT command if it precedes *:Label*.

If EOF occurs before *:Label* is found, processing terminates.

Example

```
BUILDDIM 2 "NEWGENS.RUL" 2 "NEWGENS.TXT" 4 "REJREC.ERR";
GOTO "NEWTARGET";          /* Forced branch */
LOADDATA 2 "JANACT.TXT";   /* Skip LOADDATA */
:NEWTARGET;                /* Move here */
EXIT;                      /* and exit */
```

IFERROR

Checks the status returned by a command and either continues processing or branches to the associated label in response to the status.

Syntax

```
IFERROR "Label"; <SKIPPED COMMANDS> :Label ; <COMMANDS OR EOF>
```

Parameter Description

"Label" String of characters terminated by a whitespace; not case-sensitive.

:Label Target location, preceded by a colon (:), and associated with "Label". Processing skips to this label.

Notes

This command provides the functionality of error checking and conditional branching on errors.

If the previously executed command returned a nonzero status, processing skips all subsequent commands and moves to the associated label, where it resumes. Commands that follow :Label can implement error handling or stop processing.

Processing ignores even the EXIT command if it precedes :Label. If EOF occurs before :Label is found, processing terminates.

Example

```
LOGIN "IRIS" "SYS" "PASSWORD";
SELECT "DANI" "TEST";
BUILDDIM 2 "NEWGENS.RUL" 2 "NEWGENS.TXT" 4 "REJREC.ERR";
IFERROR "DIMBUILDFAILED"; /* If BUILDDIM fails */
LOADDATA 2 "JANACT.TXT";  /* Skip LOADDATA */
:DIMBUILDFAILED;         /* Move here */
EXIT;                    /* and exit */
```

IMPORT

Loads data values from an external source into the currently selected database.

Syntax

```
IMPORT numeric dataFile fileType y/n ruleLoc rulobjName y/n [ErrorFile]
```

For an SQL data source, the syntax is as follows:

```
IMPORT 4 SQLUserName SQLUserPassword Ruleloc rulobjName y/n [ErrorFile]
```

Parameter	Description
numeric	Location of the <i>dataFile</i> file. Values: 1 - Local/client data file. 2 - Remote/server data file. 3 - File. 4 - SQL source.
dataFile	Name of data source file.
fileType	File type of <i>dataFile</i> . Values: 1 - Excel file 2 - Lotus 2 file (No longer supported) 3 - Lotus 3 file (No longer supported) 4 - Text file (No longer supported) 5 - Lotus 4 file (No longer supported)
y/n	Whether to use rules when importing <i>dataFile</i> .
ruleLoc	Location of the <i>ruleObjName</i> file. Values: 1 - Local/client rule object file 2 - Remote/server rule object file 3 - File. Use option 3 if the file is not an Essbase object, or if you want to specify the full path name. Otherwise, Essbase looks in the <APPNAME>/<DBNAME> directory.
ruleObjName	Name of the rules file.
y/n	Whether to abort on error.
SQLUserName	User name that connects to the SQL database.
SQLUserPassword	User password for the SQL database..
ErrorFile	The name of the error file. This is required only if you choose not to abort on error.

Notes

- Use the LOADDATA or UPDATEFILE commands to load data without a rules file.
- Use the BUILDDIM command to build one or more dimensions in an outline.

Example

Example 1

```
IMPORT 2 "ACTUALS" 4 "Y" 2 "ACTUALS" "Y";
```

Example 2

The following UNIX example imports from an SQL data source, and specifies an error file.

```
import 4 "tbc" "password" 2 "sales" "N" /app1/imperror;
```

The following Windows example does the same as the above.

```
import 4 "tbc" "password" 2 "sales" "N" "c:\valscrt.ERR";
```


See Also

- [LOADDATA](#)
- [UPDATEFILE](#)
- [BUILDDIM](#)

INCBUILDDIM

Build one or more dimensions from a data file, without restructuring the database. This command is designed to be used when building an outline from multiple data sources. You can save time by deferring restructure. Deferred-restructure dimension building is also called incremental dimension building.

Syntax

```
INCBUILDDIM location rulobjName dataLoc sourceName fileType errorLog appendLog
```

Parameter	Description
-----------	-------------

location	Location of the rules file. Values: <ul style="list-style-type: none">1 - Local/client-based rules file2 - Remote/server rules file3 - File. Use option 3 if the file is not an Essbase object, or if you want to specify the full path name. Otherwise, Essbase looks in the <APPNAME>/<DBNAME> directory.
rulobjName	Name of the rules file.
dataLoc	Location of the data file. Values: <ul style="list-style-type: none">1 - Local/client data file2 - Remote/server data file3 - File. Use option 3 if the file is not an Essbase object, or if you want to specify the full path name. Otherwise, Essbase looks in the <APPNAME>/<DBNAME> directory.4 - SQL source
sourceName	Source of the data file. Values: <ul style="list-style-type: none">● If <i>dataLoc</i> is 1 or 2, specify the data file name.● If <i>dataLoc</i> is 3, specify the data file name and path.● If <i>dataLoc</i> is 4, specify the SQL user name and password.
fileType	Data file type. Values: <ul style="list-style-type: none">1 - Excel file2 - Lotus .WK1 file (No longer supported)3 - Lotus .WK3 file (No longer supported)4 - Text file5 - Lotus .WK4 file (No longer supported). <p>This parameter is not required if you are using an SQL source.</p>
errorLog	Name of text file to receive error messages and rejected records. Each INCBUILDDIM command in a BEGININCBUILDDIM...ENDINCBUILDDIM block can specify a different error log.

Parameter Description

appendLog	Specifies whether to append to the error log file or overwrite it. Values: 1 - Append 2 - Overwrite
verify	Parameter specifying whether to verify the outline resulting from the deferred-restructure dimension build. Values: Y - Yes, verify the outline. This is the default. N - No, do not verify the outline.

Notes

Use a INCDIMBUILD command for each data source and rules file to be included in the dimension build. Use a BEGININCBUILDDIM command at the beginning of a group of INCDIMBUILD commands. Use an ENDINCBUILDDIM command at the end of the group of INCBUILDDIM commands.

The INCBUILDDIM command changes dimensions in the .OTN file according to the specified rules file and data file. See [BEGININCBUILDDIM](#) for information on the .OTN file.

Each rules file can build one or more dimensions. If a rules file builds multiple dimensions and an error occurs in a record for any dimension, Essbase rejects the entire record. As a result, other dimensions represented in that record might not build correctly. Consider designing dimension builds with multiple rules files using INCBUILDDIM.

An example of this problem relates to the Add as Child build method. Break the rules file into multiple rules files if both of the following circumstances apply:

- The rules and data files specify more than one Add as Child member per record.
- One of the members being added already exists in the outline as a child of any other parent.

Consider, for example, adding Mbr1 and Mbr2 as children of Par1 and Par2:

```
Par1     Par2  
  Mbr1    Mbr2
```

If Mbr1 already exists in the outline as the child of some other parent than Par1, you need to break the rules file into two separate builds. Otherwise, when Essbase sees that the member already exists in the outline, it rejects the entire record.

By default, each step of a deferred-restructure dimension build must produce a valid outline. You can use the *verify* N parameter to create an interim outline that is not valid and then update the outline in a subsequent INCBUILDDIM command to ensure the outline is valid. To verify the outline in a subsequent INCBUILDDIM command, remove the *verify* parameter or specify a Y. **Make sure that the last INCBUILDDIM command verifies the outline.**

INCBUILDDIM is identical to BUILDDIM, except for the following:

- INCBUILDDIM does not automatically restructure the database after modifying the dimensions. You can have several consecutive INCBUILDDIM commands inside a BEGININCBUILDDIM...ENDINCBUILDDIM block. Essbase restructures when it encounters ENDINCBUILDDIM.

- INCBUILDDIM enables you to append to, rather than overwrite, the error log.
- BUILDDIM does not enable you to bypass outline verification.

Example

Example 1

The following command builds the dimensions specified in GENREF.RUL and LEVELMUL.RUL, discards all data, and saves the new outline after the dimension builds are complete:

```
BEGININCBUILDDIM;
  INCBUILDDIM 2 "GENREF.RUL" 2 "GENREF.TXT" 4 "ERR.OUT" 1 "N";
  INCBUILDDIM 2 "LEVELMUL.RUL" 2 "LEVELMUL.TXT" 4 "ERR.OUT" 1 "Y";
ENDINCBUILDDIM 4;
```

Note that you can use the same rules file with multiple data files, providing the data files conform to the formatting and rules saved in the rules file. For example:

```
BEGININCBUILDDIM
  INCBUILDDIM 2 "GENREF.RUL" 2 "GENREF1.TXT" 4 "ERR.OUT" 2 "N";
  INCBUILDDIM 2 "GENREF.RUL" 2 "GENREF2.TXT" 4 "ERR.OUT" 1 "N";
  INCBUILDDIM 2 "GENREF.RUL" 2 "GENREF3.TXT" 4 "ERR.OUT" 1 "Y";
ENDINCBUILDDIM 4;
```

Example 2

The following Windows example imports dimensions from a server based text file, using a server based rules file, and specifies an error file.

```
INCBUILDDIM 2 "Genref.rul" 2 "Genref.txt" 4 "c:\valscrt.ERR" 2;
```

See Also

- [BUILDDIM](#)
- [BEGININCBUILDDIM](#)
- [ENDINCBUILDDIM](#)

LISTALIASES

Returns a list of alias tables that are defined for the currently selected database.

Syntax

```
LISTALIASES
```

Example

```
LISTALIASES;
```

LISTAPP

Returns a list of applications that are defined on the Essbase Server.

Syntax

```
LISTAPP
```

Example

```
LISTAPP;
```

LISTDB

Returns a list of databases defined on the currently selected application.

Syntax

```
LISTDB
```

Example

```
LISTDB;
```

LISTFILES

Helps track disk space used by Essbase databases by supplying accurate index and data file information.

Syntax

```
LISTFILES fileType appName dbName
```

Parameter Description

fileType Type of file for which to display information. Values:

1. Index files.
2. Data files.
3. Index and data files. This is the default.

appName Name of the application for which information is requested. Required only if no application is selected.

dbName Name of the database for which information is requested. Required only if no database is selected.

Notes

The LISTFILES command provides index and data file names, counts, sizes, and totals, and indicates whether each file is presently opened by Essbase. The file size information provided by LISTFILES is accurate, whereas the information provided by the Windows operating system for index and data files on NTFS volumes may not be accurate.

Example

```
LISTFILES;
```

Returns:

```
----- Index File Information -----
```

```
Index File Count:      1

File 1:
  File Name:           C:\Hyperion\products\Essbase\EssbaseServer\app\Sample\Basic
  \ess00001.ind
  File Type:           INDEX
  File Number:         1 of 1
  File Size:           8,024 KB (8,216,576 bytes)
  File Opened:        Y

Index File Size Total: 8,024 KB (8,216,576 bytes)
```

----- Data File Information -----

```
Data File Count:      1

File 1:
  File Name:           C:\Hyperion\products\Essbase\EssbaseServer\app\Sample\Basic
  \ess00001.pag
  File Type:           DATA
  File Number:         1 of 1
  File Size:           8,008 KB (8,200,192 bytes)
  File Opened:        Y

Data File Size Total: 8,008 KB (8,200,192 bytes)

File Size Grand Total: 16,032 KB (16,416,768 bytes)
```

LISTFILTERS

Lists the filters in a database.

Syntax

```
LISTFILTERS appName dbName
```

Parameter Description

appName Name of the application containing the filters.

dbName Name of the database containing the filters.

Example

```
LISTFILTERS "FINANC95" "SALES95";
```

LISTGROUPS

Returns a list of user groups that are defined on the Essbase Server.

Syntax

```
LISTGROUPS
```

Example

```
LISTGROUPS;
```

LISTGROUPUSERS

Returns a list of users that belong to a specified group.

Syntax

```
LISTGROUPUSERS groupName
```

Parameter	Description
-----------	-------------

<i>groupName</i>	Name of the group for which to return a list of users.
------------------	--------------------------------------------------------

Example

To return a list of all users that belong to the group called MARKETING:

```
LISTGROUPUSERS "MARKETING";
```

LISTLINKEDOBJECTS

Lists information about the objects linked to the active database for a given user name or modification date.

Syntax

```
LISTLINKEDOBJECTS userName modDate
```

Parameter	Description
-----------	-------------

<i>userName</i>	The name of a user. If specified, Essbase returns a list of all objects last modified by the given user.
-----------------	----------------------------------------------------------------------------------------------------------

<i>modDate</i>	A modification date. If specified, Essbase returns a list of all objects modified on or before the given date.
----------------	----------------------------------------------------------------------------------------------------------------

Notes

This command lists information about linked objects, including the object type, name, and description, based on criteria you specify. If you specify both a user name and modification date, objects matching both criteria are listed. If you specify no user name or date, a list of all linked objects in the database appears.

You must select a database before using LISTLINKEDOBJECTS.

For more information on linked objects, see the *Oracle Essbase Database Administrator's Guide*.

Example

To list all objects last modified by user Diana on or before July 7, 1997:

```
LISTLINKEDOBJECTS "Diana" "07/07/1997";
```

LISTLOCATIONS

Displays all location aliases defined on the current database.

Syntax

```
LISTLOCATIONS
```

Notes

This command displays the location alias parameters as defined and created with the [CREATELOCATION](#) command. You must have at least Database Manager permission to list location aliases.

Example

```
LISTLOCATIONS;
```

Returns:

Location Alias	Server	Application	Database	Username
Alias4	Aspen	Sample	Interntl	admin
Alias3	Aspen	Demo	Basic	user1
Alias2	Aspen	Samppart	Company	partitionuser
Alias1	Aspen	Sample	Basic	Admin

See Also

- [CREATELOCATION](#)
- [DELETELOCATION](#)

LISTLOCKS

Returns a list of all users who have locks on blocks for the currently selected database.

Syntax

```
LISTLOCKS [appName dbName]
```

Parameter	Description
-----------	-------------

appName dbName Optional. Both parameters required if no application and database are selected.

Example

```
LISTLOCKS;
```

LISTLOGINS

Returns the list of login instances in a session.

Syntax

```
LISTLOGINS
```

Example

The following interactive example uses LISTLOGINS to get information needed for a subsequent SETLOGIN command. Commands typed by the user are shown in **bold**.

```
localhost:::system[1]->listlogins  
ListLogins:
```

```
There are 2 Active Login Sessions.  
  Login Session 1 -- localhost  system  
  Login Session 2 -- localhost  EWhite
```

```
localhost:::system[1]->setlogin 2  
SetLogin:
```

```
Switch to Login Session 2 -- localhost  EWhite
```

LISTOBJECTS

Returns a list of objects.

Syntax

```
LISTOBJECTS number appName dbName
```

Parameter Description

number Type of object to list. Values:

- 0 - Abort
- 1 - Outline object
- 2 - Calculation script
- 3 - Report script
- 4 - Rules object
- 5 - Alias table
- 6 - Structure file
- 7 - Backup file
- 8 - Worksheet of any type
- 9 - Text object
- 10 - Partition
- 11 - Linked Reporting Object (stored)
- 12 - Selection
- 13 - Wizard

appName Name of the application containing the objects.

dbName Name of the database containing the objects.

Notes

- The list of objects returned by the LISTOBJECTS command includes object names and the status of object locks.
- Two values for the *objType* parameter, 6 and 7, are retained only for backward compatibility with Release 2.0.

- Option 11, Linked Reporting Object, lists only stored LROs; that is, files with the .LRO extension. It does not list URLs, cell notes, or linked partitions. Use the LISTLINKEDOBJECTS command to list these objects.

Example

To return a list of outline objects associated with the BASIC database:

```
LISTOBJECTS 1 "SAMPLE" "BASIC";
```

LISTUSERS

Returns a list of the users that are defined on the Essbase Server.

Syntax

```
LISTUSERS
```

Example

```
LISTUSERS;
```

LISTVARIABLES

Lists all existing substitution variables and their corresponding values for a specified Essbase Server, application, or database.

Syntax

```
LISTVARIABLES serverName [appName [dbName]]
```

Parameter	Description
-----------	-------------

<i>serverName</i>	Name of the Essbase Server host computer on which the variable is defined.
-------------------	----------------------------------------------------------------------------

<i>appName</i>	Optional. Name of the application for which the variable is defined.
----------------	----------------------------------------------------------------------

<i>dbName</i>	Optional. Name of the database for which the variable is defined.
---------------	-------------------------------------------------------------------

Example

```
LISTVARIABLES "Bamboo" "Sample" "Basic";
```

LOADALIAS

Loads an alias table to the currently selected database.

Note: See the *Oracle Essbase Database Administrator's Guide* for more information about alias tables in a database.

Syntax

```
LOADALIAS aliasName fileName
```

Parameter Description

aliasName Name of the alias table to load.

fileName Name of the data source file that loads into the table. The source file must be located on the on the Essbase Server computer, not a client computer. Specify the file name in either of the following ways:

- Full path to source file on the Essbase Server computer; for example,
C:\Hyperion\products\Essbase\EssbaseServer\app\Sample\Basic\seasonal.txt
- Relative path to the app\db directory on the Essbase Server computer; for example,
sample\basic\seasonal.txt

The data in the file must be formatted correctly. See the *Oracle Essbase Database Administrator's Guide* for details.

Example

Assume that `seasonal.txt` is a file with the following contents:

```
$ALT_NAME  
"400-10"      Guava  
"400-20"      Tangerine  
"400-30"      Mango  
$END
```

To load the contents of the `seasonal.txt` data source file into the alias table called `special_flavors`, use the following command:

```
LOADALIAS "special_flavors" "C:\Hyperion\products\Essbase\EssbaseServer\app\Sample\Basic\seasonal.txt";
```

LOADAPP

Loads an application and its respective databases into memory.

Syntax

```
LOADAPP appName
```

Parameter Description

appName Name of the application to load.

Notes

This command loads an application and databases into memory. In order for users to access information in databases, the application or individual database must be loaded into memory.

Example

To load an application called `Sample` into memory:

```
LOADAPP "Sample";
```

LOADDATA

Loads data without a rules file.

Syntax

```
LOADDATA numeric fileName
```

Parameter Description

numeric Location of the data file. Values:

1 - Local/client-based rules file (file).

2 - Remote/server data file.

3 - File. Use option 3 if the file is not an Essbase object, or if you want to specify the full path name. Otherwise, Essbase looks in the <APPNAME>/<DBNAME> directory.

Note: Essbase Servers installed on Windows computers can accept a spreadsheet file (.xls) using option 3; Essbase Servers installed on UNIX computers cannot accept spreadsheet files.

fileName Name of the file to load.

Example

```
LOADDATA 2 "calcdat";
```

LOADDB

Loads a database into memory.

Syntax

```
LOADDB appName dbName
```

Parameter Description

appName Name of the application in which the database resides.

dbName Name of the database to load.

Notes

This command loads a database into memory. A database must be loaded into memory in order for users to access its information.

Example

To load a database called BASIC from an application called SAMPLE:

```
LOADDB "SAMPLE" "BASIC";
```

LOGIN

Connects the current ESSCMD session to Essbase Server.

Syntax

```
LOGIN hostNode userName password [appName dbName]
```

Parameter Description

hostNode Host name of the Essbase Server computer.

userName User ID defined on the Essbase Server.

password User's password.

appName Optional. Name of the application to load.

dbName Optional. Name of the database to load.

Notes

- The Essbase Server must already be running before a login can occur.
- If you want to use the optional *appName* and *dbName* parameters, you must use both.
- With the optional parameters, this command is the equivalent of logging in and issuing a `SELECT appName and dbName` command.

Example

To log in a user named TomT who is using ESSCMD from the Essbase Server computer:

```
LOGIN "LOCAL" "TOMT" "PASSWORD";
```

To log in a user named TomT to a remote Essbase Server on a host named BEECH:

```
LOGIN "BEECH" "TOMT" "PASSWORD";
```

LOGOUT

Logs the current ESSCMD user off from the Essbase Server.

Syntax

```
LOGOUT
```

Notes

This command logs the current ESSCMD user off from the Essbase Server, but does not exit the ESSCMD session.

Example

```
LOGOUT;
```

LOGOUTALLUSERS

Logs off all users from the Essbase Server.

Syntax

```
LOGOUTALLUSERS Y|N
```

Parameter Description

Y|N Sets whether users are logged out.

Notes

This command logs out all users except for the user issuing the command.

Example

```
LOGOUTALLUSERS "Y";
```

See Also

- [LOGOUTUSER](#)

LOGOUTUSER

Logs a specified user off the Essbase Server.

Syntax

```
LOGOUTUSER userNumber
```

Parameter Description

userNumber Login ID number associated with a user. Issue LOGOUTUSER with no parameter to display a list of users and user numbers.

Notes

- This command is available in interactive mode only.
- To find the user number, issue this command without a parameter. ESSCMD displays a list of logged-in users with numbers representing their login order. You can select the user to log off.

Example

To log the user whose user number is 1 off the Essbase Server:

```
LOGOUTUSER 1;
```

See Also

- [LOGOUTALLUSERS](#)

OUTPUT

Directs process information output from the ESSCMD session to a text file.

Syntax

```
OUTPUT outputType [outputName] / [errorName]
```

Parameter	Description
-----------	-------------

<i>outputType</i>	Number representing output operation. Values:
-------------------	-----------------------------------------------

- 1 - Outputs all process information.
- 2 - Outputs only errors.
- 3 - Stops output of process information.
- 4 - Stops output of errors.

<i>outputName</i>	Required for <i>outputType</i> 1 only. Name of file to receive output. Not used with other values for <i>outputType</i> .
-------------------	---------------------------------------------------------------------------------------------------------------------------

<i>errorName</i>	Required for <i>outputType</i> 2 only. Name of file to receive errors. Not used with other values for <i>outputType</i> .
------------------	---------------------------------------------------------------------------------------------------------------------------

Notes

This command directs Essbase to send messages from the ESSCMD session to the specified file instead of to the screen.

Example

To write statistics tables returned from the [GETPERFSTATS](#) command to a text file called "stats":

```
OUTPUT 1 "stats"; :Send process info from ESSCMD to file "stats"  
GETPERFSTATS; :Execute this command  
OUTPUT 3 "stats"; :Stop sending process info to file "stats"
```

Result: Essbase writes performance statistics to the file "stats" instead of to the screen.

To write errors during the session to a file called CMDERR:

```
OUTPUT 2 "CMDERR";
```

To write statistics to the output file STATINFO:

```
OUTPUT 1 "STATINFO";
```

To write only the information that the calculation ran, and not all messages:

```
OUTPUT 1 "CALCDEFAULT";
```

PAREXPORT

Starts the parallel data-export process.

The export process does not begin until all users are logged out of the database. After the export process begins, the database is in read-only mode. Users can read the data but they cannot change

it. After the export process is finished, Essbase returns the database to read-write mode and users can make changes to the data.

Syntax

```
PAREXPORT [-threads n] [-in input_filename] | [output_filename] amount formatOption
```

Parameter	Description
-threads n	Overrides the default number of export threads set in the EXPORTTHREADS setting in the <code>essbase.cfg</code> file. The maximum value is 8. If <i>n</i> is greater than 8, Essbase assumes the value to be 8.
-in input_filename	Specifies the full path name of an input file that contains a list of export file names. The number of files listed in the input file must match the number of export threads. Parallel export gracefully errors out if there is a mismatch. <ul style="list-style-type: none">● If the data for any export thread exceeds 2 GB, Essbase creates additional files, none of which exceeds 2 GB. See Note for details.● If <code>-in</code> is not specified, the next value is assumed to be the value of the <code>output_filename</code> parameter.● If the listed files in the input file do not include a path, the files are created in the <code>ARBORPATH\app</code> directory.
output_filename	Specifies the path and root for the file names created to contain the export data. For each thread, a number is appended to the specified <code>output_filename</code> . For example, if <code>outfile_filename</code> is <code>outfile</code> and two threads are specified, the resulting file names are <code>outfile1</code> and <code>outfile2</code> . If the data for a thread exceeds 2 GB, that export data is divided into multiple files with a second number appended to the file names. See Note for details. If no path is specified, the file is created in the <code>ARBORPATH\app</code> directory.
amount	Specifies the number representing the data to export. <ul style="list-style-type: none">● 1 - All data● 2 - Only level 0 blocks● 3 - Only data from blocks with input data
formatOption	Specifies the format of the data. <ul style="list-style-type: none">● 0 (null) - Non-columnar format. This is the default.● 1 - Columnar format

Notes

- With this command, users can override the default number of export threads specified in the EXPORTTHREADS setting, and they can provide a list of export file names. During the export process, multiple threads can retrieve data and write to their corresponding export files concurrently.
- Parallel export creates multiple export files based on the number of export threads specified. The database is divided as evenly as possible among the number of parallel export threads.
- If the data for an export thread exceeds 2 GB, that data is separated into multiple files. Each file is less than 2 GB. The first file name retains the original name; Essbase appends `_1`, `_2`, and so on, as needed, to the additional files.

- The `PARAEXPORT` command works on both aggregate storage and block storage databases, however aggregate storage exports work differently from block storage exports. See the *Oracle Essbase Database Administrator's Guide* for more information.

Example

```
PARAEXPORT -threads 4 -in e:\data\input.txt 1 1;
```

Note that `e:\data\input.txt` is a text file that contains four file names on separate lines; that is,

```
e:\data\export1.txt
```

```
e:\data\export2.txt
```

```
d:\data\export3.txt
```

```
d:\data\export4.txt
```

In this example, all data in the database is divided among four export threads to create four export files. The data is exported in columnar format.

If the data intended for a file is greater than 2 but less than 4 GB, Essbase creates two files. For example, for the data apportioned to `e:\data\export2.txt`, Essbase would create `e:\data\export2.txt` and `e:\data\export2_1.txt`.

See Also

- [EXPORT](#)
- [“EXPORTTHREADS” on page 444](#)

PRINTPARTITIONDEFFILE

Produces a text file of the partition-mapping tables of the distributed database.

Syntax

```
PRINTPARTITIONDEFFILE location [ddbFileName] textFileName
```

Parameter	Description
<code>location</code>	Possible values: 1- Local/client file with a <code>.DDB</code> file extension that is stored in the directory pointed to by <code>ARBORPATH</code> . The <code>ddbFileName</code> is automatically retrieved. 2- Remote/server <code>.DDB</code> file. The <code>ddbFileName</code> is automatically retrieved. 3- Local/client file not stored in the <code>ARBORPATH</code> , or without a <code>.DDB</code> file extension. The <code>ddbFileName</code> is required when using this option.
<code>ddbFileName</code>	The name of the partition mapping definition <code>.DDB</code> file from which to read information. This is usually the name of the database; for example, <code>BASIC.DDB</code> . If <code>location</code> is 1 or 2, <code>ddbFileName</code> is not required. If <code>location</code> is 3, the full path, file name, and file extension of the file is required.
<code>textFileName</code>	The full path, file name, and file extension of the text output file to create.

Notes

This command produces a text file of the partition-mapping tables of the distributed database. The file contains the following information for each partition:

- Total number of partitions
- Partition host, application, database, and user
- Time the partition was last modified
- Partition definition
- Connection information
- Partition shape definition
- Partition type information
- Database map information
- Slice map information
- Region identification
- Outline change direction

Example

```
PRINTPARTITIONDEFFILE "2" "basic.txt";
```

PURGELINKEDOBJECTS

Deletes objects linked to the active database for a given user name or modification date.

Syntax

```
PURGELINKEDOBJECTS userName modDate
```

Parameter Description

userName The name of a user. If *userName* is specified, Essbase deletes all objects last modified by the given user.

modDate A modification date. If *modDate* is specified, Essbase deletes all objects modified on or before the given date.

Notes

This command deletes linked objects based on criteria you specify. A list of the objects matching your criteria appears as they are being deleted. If you specify both a user name and modification date, objects matching both criteria are deleted. If you specify no user name or date, all linked objects in the database are deleted.

You must select a database before using PURGELINKEDOBJECTS. You must also have design privilege for the database to delete any objects.

For more information on linked objects, see the *Oracle Essbase Database Administrator's Guide*.

Example

To delete all objects last modified by user Diana on or before July 7, 2002:

```
PURGELINKEDOBJECTS "Diana" "07/07/2002";
```

PURGEOTLCHANGEFILE

Deletes outline changes that already have been applied from the .CHG log file.

Syntax

```
PURGEOTLCHANGEFILE serverName appName dbName partitionType direction
```

Parameter	Description
-----------	-------------

<i>serverName</i>	Name of the computer hosting the Essbase Server from which to delete .CHG information.
-------------------	----------------------------------------------------------------------------------------

<i>appName</i>	Name of the application from which to delete .CHG information.
----------------	----------------------------------------------------------------

<i>dbName</i>	Name of the database from which to delete .CHG information.
---------------	-------------------------------------------------------------

<i>partitionType</i>	Name of the partition type to which the deletions are applied: 1 - Replicated. 2 - Linked. 3 - Transparent.
----------------------	----------------------------------------------------------------------------------------------------------------------

<i>direction</i>	Values:
------------------	---------

- Source - The selected database is used as a data source for the replicated, transparent, or linked partition.
- Target - The selected database is used as a data target for the replicated, transparent, or linked partition.

Example

```
PURGEOTLCHANGEFILE "BAMBOO" "Sample" "Basic" "1" "Source";
```

PUTALLREPLCELLS

Replicates all data cells in a replicated partition from the data source database you selected with the SELECT command, to a specified data target database. Use this command when you are in the data source database.

Syntax

1:

```
PUTALLREPLCELLS targetServerName targetAppName targetDbName
```

2:

```
PUTALLREPLCELLS ALL
```

Parameter	Description
targetServerName	Host name of the computer where the data target resides.
targetAppName	Name of the data target application to which cells are replicated.
targetDbName	Name of the data target database to which cells are replicated.
ALL	Updates all cells in partitions where the selected database is a data replication source.

Notes

PUTALLREPLCELLS command puts all replicated data cells from the selected data source and replicates them to the data target database. This is useful when the data in the source and target databases are out of synch and need to be resynchronized.

PUTALLREPLCELLS puts cells from the data source server to the data target server, based on a request made from the data source; [GETALLREPLCELLS](#) gets cells from the data source to the data target, based on a request made from the data target.

Example

```
PUTALLREPLCELLS "Aspen" "Sample" "Basic";
```

See Also

- [GETALLREPLCELLS](#)
- [PUTUPDATEDREPLCELLS](#)

PUTUPDATEDREPLCELLS

This command replicates all changed data cells in the replicated partition from the data source database you selected with the SELECT command, to the specified data target database. Use this command when you are in the data source database.

Syntax

1:

```
PUTUPDATEDREPLCELLS targetServerName targetAppName targetDbName
```

2:

```
PUTUPDATEDREPLCELLS ALL
```

Parameter	Description
targetServerName	Host name of the computer where the data target resides.
targetAppName	Name of the data target application to which changed cells are replicated.
targetDbName	Name of the data target database to which changed cells are replicated.
ALL	Updates all changed cells in all partitions where the selected database is a data replication source.

Notes

The PUTUPDATEDREPLCELLS command takes all changed replicated data cells from the selected data source, and replicates them in the data target database.

Essbase determines what updates are performed based on an internal time stamp which is read at the block level. Whenever data in the block changes, Essbase updates the time stamp to the current time. If data is changed that is not defined in the replication area, but is part of the data block, the time stamp is still reset. Therefore, it is possible to update data in the replication area, even though the replicated data has not changed.

When a block is removed by such actions as RESETDB, and you request an update of the replication cells, Essbase performs an internal search that identifies blocks without time stamps. Essbase then gets all cells from the replication area, instead of only changed cells. This may take some time, depending on the size of the block.

PUTUPDATEDREPLCELLS puts cells from the data source server to the data target server, based on a request made from the data source; GETUPDATEDREPLCELLS gets cells from the data source to the data target, based on a request made from the data target.

Example

```
PUTUPDATEDREPLCELLS "Aspen" "Sample" "Basic";
```

See Also

- [GETUPDATEDREPLCELLS](#)
- [PUTALLREPLCELLS](#)

REMOVELOCKS

Removes any locks that a specified user has acquired through a spreadsheet operation.

Syntax

```
REMOVELOCKS userNumber
```

Parameter	Description
-----------	-------------

<i>userNumber</i>	Login ID of the user for whom you are removing locks.
-------------------	-------------------------------------------------------

Notes

This command removes locks acquired through a spreadsheet operation. Removing locks is sometimes required for maintenance-related activities. Removing a user's lock forces a logout of that user's session. To display the list of users who have locks, use [LISTLOCKS](#).

Example

To remove all locks that are held by user number 1 on the currently selected database:

```
REMOVELOCKS 1;
```

REMOVEUSER

Removes a user from a group.

Groups are used to classify users with identical security requirements.

Syntax

```
REMOVEUSER groupName userName
```

Parameter	Description
-----------	-------------

<i>groupName</i>	Name of group from which to remove user.
------------------	------------------------------------------

<i>userName</i>	Name of the user to remove.
-----------------	-----------------------------

Notes

If you want to completely delete a user from Essbase, use the [DELETEUSER](#) command. Deleting the user ID deletes the user from the list of users on the Essbase Server, as well as logging the user out of the active session.

Example

To remove the user DANTE from the group called INTERNTL:

```
REMOVEUSER "INTERNTL" "DANTE" ;
```

See Also

- [DELETEDGROUP](#)
- [DELETEUSER](#)

RENAMEAPP

Renames an application.

Syntax

```
RENAMEAPP sourceApp newAppName
```

Parameter	Description
-----------	-------------

<i>sourceApp</i>	Name of existing application.
------------------	-------------------------------

<i>newAppName</i>	New name for application.
-------------------	---------------------------

Example

```
RENAMEAPP "FINANC95" "ANNFIN95" ;
```

RENAMEDB

Renames a database.

Syntax

```
RENAMEDB sourceApp sourceDb newDbName
```

Parameter	Description
-----------	-------------

<i>sourceApp</i>	Name of the application that contains the database to be renamed.
------------------	-------------------------------------------------------------------

<i>sourceDb</i>	Name of the database to be renamed.
-----------------	-------------------------------------

<i>newDbName</i>	New name for the database.
------------------	----------------------------

Example

```
RENAMEDB "FINANC95" "SALES95" "95SALES";
```

RENAMEFILTER

Renames a filter.

Syntax

```
RENAMEFILTER sourceApp sourceDb sourceFltr newFltrName
```

Parameter	Description
-----------	-------------

<i>sourceApp</i>	Name of the application that includes the filter.
------------------	---------------------------------------------------

<i>sourceDb</i>	Name of the database that includes the filter.
-----------------	------------------------------------------------

<i>sourceFltr</i>	Name of the existing filter.
-------------------	------------------------------

<i>newFltrName</i>	New name for filter.
--------------------	----------------------

Example

```
RENAMEFILTER "FINANC95" "SALES95" "FILTER95" "95FILT";
```

RENAMEOBJECT

Renames an existing object.

Syntax

```
RENAMEOBJECT objType sourceApp sourceDb sourceObj newObjName
```

Parameter	Description
objType	Type of object to rename. Values: 0 - Abort 1 - Outline object, not available 2 - Calculation script 3 - Report script 4 - Rules object 5 - Alias table 6 - structure file 7 - Backup file, not available 8 - Worksheet of any type, not available 9 - Text object 10 - Partition 11 - Selection 12 - Wizard
sourceApp	Name of the application that includes the object.
sourceDb	Name of the database that includes the object.
sourceObj	Name of the existing object.
newObjName	New name for the object.

Notes

Two values for the *objType* parameter, 6 and 7, are retained only for backward compatibility with Release 2.0.

Example

```
RENAMEOBJECT 2 "FINANC95" "SALES95" "OLDOBJ" "ARCHIVE";
```

RENAMEUSER

Renames a user.

Syntax

```
RENAMEUSER userName newUserName
```

Parameter	Description
userName	Name of the existing user.
newUserName	New name for the user.

Notes

To rename a user, you must have at least Create/Delete User permission.

Example

```
RENAMEUSER "NEWUSER" "D_ROSETTI";
```

REPORT

Executes one or more report strings.

Syntax

```
REPORT reportString
```

Parameter	Description
-----------	-------------

reportString	One or more report strings.
--------------	-----------------------------

Notes

When working with ESSCMD in interactive mode, use this command to enter one or more strings from a report script. Interactive ESSCMD prompts for a string each time you press the Enter key. When finished, end with a blank string.

When using the REPORT command in ESSCMD scripts, end each line with a backslash.

Example

Example of interactive use: To create a report based on all descendants of Qtr1, including the Qtr1 member, and all children of Market, including the Market member, enter the text shown in this color. In this example, ESSCMD prompts are in black. Instructions to press the Enter key are in this color.

```
local:sample:basic:admin(1)->REPORT  
Report:
```

```
Enter blank string to end report  
Enter string ><IDESCENDANTS Qtr1(Press Enter)  
Enter string ><ICHILDREN Market(Press Enter)  
Enter string >!(Press Enter)  
Enter string > (Press Enter)
```

Example of use in an ESSCMD script: To include commands in an ESSCMD script to generate the same report, end each line with a backslash.

```
Report:  
  
IDESCENDANTS Qtr1\  
ICHILDREN Market\  
!\  
\
```

See Also

- [REPORTLINE](#)

REPORTLINE

Executes a single report string.

Syntax

```
REPORTLINE reportString
```


Parameter	Description
-----------	-------------

reportString	Report string.
--------------	----------------

Example

To create a report based on all descendants of Year:

```
REPORTLINE "<DESCENDANTS YEAR !";
```

See Also

- [REPORT](#)

RESETDB

Clears all the data and LROs from the currently selected database.

Syntax

```
RESETDB
```

Example

```
RESETDB;
```

RESETOTLCHANGETIME

Changes the time on the Essbase you selected with the SELECT command, to match the time on another Essbase Server.

Syntax

```
RESETOTLCHANGETIME fromPartition toPartition
```

Parameter	Description
-----------	-------------

serverName	Name of the Essbase Server from which the time change is applied.
------------	-------------------------------------------------------------------

appName	Name of the application from which the time change is applied.
---------	----------------------------------------------------------------

dbName	Name of the database from which the time change is applied.
--------	-------------------------------------------------------------

partitionType	The name of the type of partition from which the time change is applied. Values:
---------------	----------------------------------------------------------------------------------

- 1 - Replicated
- 2 - Linked
- 3 - Transparent

direction	Values:
-----------	---------

Source - The selected database is used as a data source for the replicated, transparent, or linked partition.

Target - The selected database is used as a data target for the replicated, transparent, or linked partition.

serverName	Name of the Essbase Server to get the time change.
------------	----------------------------------------------------

Parameter	Description
appName	Name of the application to get the time change.
dbName	Name of the database to get the time change.
partitionType	The name of the type of partition the time change is applied to. Values: 1 - Replicated 2 - Transparent 3 - Linked
direction	Values: <ul style="list-style-type: none"> ● Source - The selected database is used as a data source for the replicated, transparent, or linked partition. ● Target - The selected database is used as a data target for the replicated, transparent, or linked partition.

Notes

The RESETOTLCHANGETIME command synchronizes the internal time stamps between two Essbase databases that share a partition. This time stamp is used when performing GETPARTITIONOTLCHANGES and APPLYOTLCHANGEFILE operations to synchronize the outlines. That is, to propagate changes (made during a dimension build, for example) from the outline in one database sharing a partition to the other.

Partitioned databases contain a time stamp indicating when the outline was last modified. Essbase uses the time stamp when it performs GETPARTITIONOTLCHANGES and APPLYOTLCHANGEFILE operations to synchronize the outlines. When you use GETPARTITIONOTLCHANGES, the time is stamped in one of the databases. When you use APPLYOTLCHANGEFILE, Essbase reads that time stamp and writes it TO the partition definition file (*AppName.ddb*) of the other database. The direction in which changes are propagated (data source to data target, or data target to data source) is set in the partition definition

It is not necessary to use the RESETOTLCHANGETIME command when performing GETPARTITIONOTLCHANGES and APPLYOTLCHANGEFILE operations, or as part of regular maintenance. Instead, use it as needed, to reset the time stamp on a partitioned database. For example, if two databases that share a partition reside on different server computers, and a power outage affects the time stamp on one of the databases, you can use RESETOTLCHANGETIME to re-synchronize the time stamps.

For more information, see the *Oracle Essbase Database Administrator's Guide*.

Example

```
RESETOTLCHANGETIME "BAMBOO" "SAMPLE" "BASIC" "1"
"SOURCE" "ASPEN" "SAMPLE" "BASIC" "1" "TARGET";
```

See Also

- [GETPARTITIONOTLCHANGES](#)
- [APPLYOTLCHANGEFILE](#)

RESETPERFSTATS

Resets statistics gathering for a specified persistence and scope. Each of the statistics tables available using the GETPERFSTATS ESSCMD command has a pre-defined persistence and scope. When you issue RESETPERFSTATS without parameters, statistics-gathering is reset for all of the tables.

Collecting and analyzing performance statistics can assist you in determining whether databases are in good running condition, or could use modifications to improve performance.

Depending on your database and production needs, you create a statistical measurement profile by resetting the appropriate levels of *persistence* (length of events to measure) and *scope* (granularity of the entity to measure).

Syntax

```
RESETPERFSTATS persistence scope
```

Parameter	Description
<code>persistence</code> [default=long]	<ul style="list-style-type: none">● disable Turn off performance-statistics gathering.● enable Turn on performance-statistics gathering. You might do this when you want to tune the system, change hardware configuration, or monitor I/O. The measurement begins for current processes as soon as you enable it. Any subsequent queries for statistics return measurements spanning from the time of enablement to the time of the query.● medium Reset tables that measure medium-length events:<ul style="list-style-type: none">○ kernel I/O Statistics table○ Cache Endtrans Statistics table○ Database Synchronous I/O table○ Database Asynchronous I/O table● long (default) Reset tables that measure events over the course of the entire session. Long measurements rarely need to be reset. Example: kernel Cache Statistics table.
<code>scope</code> [default=all]	<ul style="list-style-type: none">● db Reset per-database statistics tables.● server Reset per-application statistics tables.● all (default) Reset all statistics tables: for threads, databases, and applications.

Notes

This command resets to zero any previously collected statistics of a persistence shorter than or equal to the reset persistence. For example, entering RESETPERFSTATS LONG resets both long and medium statistics tables back to zero.

Example

```
RESETPERFSTATS ENABLE;  
RESETPERFSTATS MEDIUM SERVER;
```

See Also

- [GETPERFSTATS](#)

RESETSTATUS

Resets all saved status values to 0 (zero).

Syntax

```
RESETSTATUS
```

Notes

RESETSTATUS is used in ESSCMD error handling.

This command resets:

- All saved status values, including that of the previous command.
- The returned status values, as tested in IFERROR.

Example

```
RESETSTATUS;
```

RUNCALC

Runs a calculation script.

Syntax

```
RUNCALC numeric calcScript
```

Parameter Description

<code>numeric</code>	Location of the calculation script data file. Values: <ul style="list-style-type: none">1 - Local/client-based calculation script.2 - Remote/server calculation script.3 - File. Use option 3 if the file is not an Essbase object, or if you want to specify the full path name. Otherwise, Essbase looks in the <APPNAME>/<DBNAME> directory.
<code>calcScript</code>	Name of the calculation script to run.

Notes

The *numeric* parameter indicates the location of the file named by the *calcScript* parameter.

Example

To execute a calculation script object named FAM100 on the Essbase Server:

```
RUNCALC 2 "FAM100";
```

RUNREPT

Runs a report script.

Syntax

```
RUNREPT numeric reptScript outputFile
```

Parameter Description

numeric Location of the report script file. Values:

1 - Local/client-based report script.

2 - Remote/server report script.

3 - File is not an Essbase object; enter a fully qualified path to the file. Use option 3 if you want to specify the full path name. Otherwise, Essbase looks in the <APPNAME>/<DBNAME> directory.

reptScript Name of the report script to run.

outputFile Target file name for report output.

Notes

The value you enter for the *numeric* parameter tells Essbase where the file named *reportScript* resides. Use the OUTPUT command to suppress the onscreen display of the script.

Example

To execute a report script called P&L on the Essbase Server:

```
RUNREPT 2 "P&L" "P&L.out";
```

SELECT

Selects the application and database on which to focus subsequent commands.

Syntax

```
SELECT appName dbName
```

Parameter Description

appName Name of the application containing the desired database.

dbName Name of database within the selected application.

Example

To select the database called BASIC in the application called SAMPLE:

```
SELECT "SAMPLE" "BASIC";
```

SETALIAS

This command sets an alias table as the primary table for reporting and any additional alias requests.

Syntax

```
SETALIAS aliasName
```

Parameter Description

aliasName Name to set for the alias table.

Example

```
SETALIAS "Long Names";
```

SETAPPSTATE

Defines application settings.

Syntax

```
SETAPPSTATE [appName] "desc" Y/N Y/N accessLevel  
Y/N Y/N Y/N Y/N lockTimeout MaxLROFileSize;
```

Parameter Description

<i>appName</i>	Name of the application. Do not include <i>appName</i> if the active application is selected.
<i>desc</i>	Text string describing the application.
<i>Y/N</i>	Sets whether the application is loadable.
<i>Y/N</i>	Sets whether autoload occurs.
<i>accessLevel</i>	Default access level. Values: 0 - None. 1 - Read. 2 - Write. 3 - Calculate. 4 - Application Manager or Database Manager
<i>Y/N</i>	Sets whether connections can be made.
<i>Y/N</i>	Sets whether commands can be issued.
<i>Y/N</i>	Sets whether updates can occur.
<i>Y/N</i>	Sets whether security is enabled.
<i>lockTimeout</i>	Maximum number of seconds that locks can be placed on blocks by Spreadsheet Add-in users.
<i>MaxLROFileSize</i>	Maximum size, in kilobytes, for a Linked Reporting Objects (LRO) file.

Notes

- Using the semicolon statement terminator (;) is optional in ESSCMD batch files. However, it is good practice to use the terminator with this command to signal the end of the parameter list. This is especially important if you omit some of the parameters and take their default values. If not all parameters are present, and the ; is omitted, ESSCMD looks for the remaining values in the next statement in the batch file, leading to unpredictable results.
- As with many other ESSCMD commands, if you issue only the SETAPPSTATE keyword in interactive mode, ESSCMD prompts you for the other values.

Example

```
SETAPPSTATE "sample" "The application is ready"  
  "Y" "Y" 0 "Y" "Y" "Y" "Y" "3600";
```

SETDBSTATE

Defines database settings. For more options, see [SETDBSTATEITEM](#).

Syntax

```
SETDBSTATE ["appName"] ["dbName"] "desc" Y/N Y/N accessLevel  
  dataCacheSize Y/N Y/N Y/N currDb ccType 0/1 indexCacheSize  
  IndexPageSize Y/N;
```

Parameter	Description
appName	Name of the application. Do not include if the application is already selected.
dbName	Name of the database; required if <i>appName</i> is specified.
desc	Text string describing the database.
Y/N	Sets whether the database is loadable.
Y/N	Sets autoloading on or off.
accessLevel	Default access level. Values: 0 - None. 1 - Read. 2 - Write. 3 - Calculate. 4 - Database Manager.
dataCacheSize	Maximum amount of memory allocated for data cache. Default: 3145728 bytes.
Y/N	Sets whether to aggregate missing values.
Y/N	Sets whether to perform a Two-Pass calc.
Y/N	Sets whether to create blocks on equations.
currDb	Links a currency database.

Parameter	Description
ccType	Specifies the default currency type member.
0/1	Sets the conversion method. Values: 0 - Division. 1 - Multiplication.
indexCacheSize	Maximum amount of memory allocated for index cache. Default: 1048576 bytes.
indexPageSize	Maximum amount allocated for index page. Index page size is now fixed at 8192 bytes regardless of this setting.
Y/N	Enable (Y) or disable (N) data compression on disk.

Notes

- Using the semicolon statement terminator (;) is optional in ESSCMD scripts. However, it is good practice to use it to signal the end of the SETDBSTATE parameter list. This is especially important if you omit some of the parameters, accepting their default values. If not all parameters are present, and the ; is omitted, ESSCMD looks for the remaining values in the next line, leading to unpredictable results.
- If you issue only the SETDBSTATE keyword in interactive mode, ESSCMD prompts you for the other values.
- Load the required database before you run the SETDBSTATE command, then stop and restart the database for this command to take effect.

Example

The following example assumes that the application and database are already selected. Settings that you want to skip need to be represented using empty quotation marks as placeholders.

```
SETDBSTATE "Data has been updated" "Y" "Y" 4 "3000000"
"N" "Y" "N" "" "" 0 "1049000" "8192" "Y";
```

See Also

- [SETDBSTATEITEM](#)

SETDBSTATEITEM

Defines database settings by number, providing more options than SETDBSTATE.

It is most efficient to load the required database before you run the SETDBSTATE command, then stop and restart the database for the command to take effect.

Note: When changing sizes, valid size-entry units in ESSCMD are bytes (b), kilobytes (k), metabytes (m), gigabytes (g), or terabytes (t). Example: 8192b, 8k, 1m, *ng*, *nt*. If no size unit is given, the default unit is bytes (b).

Syntax

```
SETDBSTATEITEM [optionNumber] [ "appName" ] [ "dbName" ] [ "values" ]
```

Parameter	Description
-----------	-------------

optionNumber	An integer between 0 and 27, inclusive. This number corresponds to the options listed below. Enter 99 to be prompted for all options (in interactive mode).
appName	Name of the application. Omit if the application is already selected using the SELECT command.
dbName	Name of the database; required if <i>appName</i> is specified.
values	Acceptable value or values; these vary from option to option. See Values for Values Parameter

Notes

- Using the semicolon statement terminator (;) is optional in ESSCMD scripts. ; however, it is good practice to use it to signal the end of the parameter list. This is especially important if you omit some of the parameters, accepting their default values. If not all parameters are present, and the ; is omitted, ESSCMD looks for the remaining values in the next line, leading to unpredictable results.
- Items 14 and 15 (Data Compression and Data Compression Type) are effective as soon as Essbase writes blocks to disk. This command has no effect on blocks already on disk until the next time Essbase writes them.
- Items 18, 19, and 20 (Isolation level, Pre-Image Access, and Time Out) are effective the next time there are no active transactions in the database.
- Items 5, 12, 21, and 22 (Data Cache Size, Index Cache Size, Blocks Modified Before Internal Commits, and Rows to Data Load Before Internal Commit) are effective when the database is stopped and re-started.
- Item 13, Index Page Size, is no longer changeable. Input for this setting is ignored.

Values for *Values* Parameter

- **0. Abort**—Returns you to the ESSCMD command line. Use only in interactive mode.
- **1. Description**—Text string describing the database.
- **2. Allow Database to Start?**—Sets whether the database is loadable. Values: Y/N.
- **3. Start Database with Application?**—Sets autoload on or off. Values: Y/N.
- **4. Access Level**—Values:
 - 0 - None.
 - 1 - Read.
 - 2 - Write.
 - 3 - Calculate.
 - 4 - Database Manager.
- **5. Data Cache Size**—The maximum size of a buffer in memory that holds data blocks for the current operation. Default and minimum: 3145728B (3 megabytes).

- **6. Aggregate Missing Values?**—Sets whether to aggregate missing values. Values: Y/N.
- **7. Two Pass Calc When [CALC ALL]?**— Sets whether or not to perform a second calculation on formulas tagged as "Two Pass" as part of the default calculation. Values: Y/N.
- **8. Create Blocks on Equation?**—Sets whether to create blocks on equations. Values: Y/N.
- **9. Currency Database Name**—Links a currency database that you specify.
- **10. Currency Conversion Type Member** —Specifies the default currency conversion type member.
- **11. Currency Conversion Type**—Sets the conversion method. Values:
 - 0 - Division.
 - 1 - Multiplication.
- **12. Index Cache Size**—Maximum size of a memory buffer that holds index pages for the current operation. Default: 1048576 bytes (1 megabyte).
- **13. Index Page Size**—This setting is no longer changeable.
- **14. Data Compression on Disk?**—Enables (Y) or disables (N) data compression on disk.
- **15. Data Compression Type**—Values:
 - 1 - Run-Length Encoding.
 - 2 - Bitmap (the default).
- **16. Retrieval Buffer Size**—Specifies the size of the internal sorting buffer that holds extracted row data cells before they are evaluated by the **RESTRICT** or **TOP/BOTTOM** Report Writer command. Default: 10K (on 32-bit platforms), and 20K (on 64-bit platforms).
- **17. Retrieval Sort Buffer Size**—Specifies the size of the internal data sorting buffer. Default: 10K (on 32-bit platforms), and 20K (on 64-bit platforms).
- **18. Isolation Level**—Choose committed or uncommitted access to your database. Committed access provides better data integrity. Uncommitted access provides consistency with Release 4. See the *Oracle Essbase Database Administrator's Guide* for information about isolation levels. Values:
 - 1 - Committed access
 - 2 - Uncommitted access (the default)

Depending on which type of access you specify, ESSCMD prompts you for other parameters (or you can supply the values on the command line).

If you choose 1 (committed access), ESSCMD prompts for:

- Pre-image access (see item 19).
- Time Out (see item 20).

If you choose 2 (uncommitted access), ESSCMD prompts for:

- Number of blocks modified before internal commit (Default: 3000).

- A value of 0 means no implicit commit; Essbase commits blocks at the end of the transaction.
- Number of rows to data load before internal commit. (Default: 0, no implicit commit; Essbase commits blocks at the end of the transaction).
- **19. Pre Image Access?**—Valid for Committed access only. Provides users Read-only access to data blocks that are locked for the duration of another transaction. Users see the last committed data values for those data blocks. If you choose N (No), your transaction waits for the blocks to become available, or Essbase issues a time-out error. Values: Y/N. Default: N (No).
- **20. Time Out**—The length of time, in seconds, to wait to acquire a lock on data blocks that are locked by another transaction. Acceptable values are:
 - -1 - Indefinite wait.
 - 0 - Immediate access, or no wait.
 - n - A number of seconds that you specify.
- **21. Number of blocks modified before internal commit**—Default: 3000. See item 18.
- **22. Number of rows to data load before internal commit**—Default: 0. See item 18.
- **23. Add Disk Volume Definitions**—Use if you want to allocate storage across multiple volumes, or restrict space used on a volume. For information on disk volumes, see the *Oracle Essbase Database Administrator's Guide*.

ESSCMD prompts you for the following values, unless you supply them on the command line:

- The number of new disk volumes you want to add.

Then, for each volume:

- Volume name or drive letter (required).
- Volume size (maximum space to use on that volume). Default: Unlimited (0). Minimum: 8 megabytes. You can specify this value in bytes (B), kilobytes (K), megabytes (M, the default), or gigabytes (G).
- File types to be stored on this volume:
 - 1 - Index files only.
 - 2 - Data files only.
 - 3 - Index and data files (the default).
- File size: the maximum size that each index or data file can attain before Essbase creates a new file. Default: 2G. Minimum: 8 megabytes. You can specify this value in bytes (B), kilobytes (K), megabytes (M, the default), or gigabytes (G).
- **24. Modify Disk Volume Information**—Change the disk volume settings on an allocated volume. This command prompts you for the number assigned to the disk volume you want to change and then prompts you for each value for the chosen disk volume. See item 23. Use [GETDBSTATE](#) to see a list of the currently defined disk volumes, and the number assigned to each volume.

- **25. Delete Disk Volume Definition**—Stop Essbase from storing additional files on an allocated volume. This command prompts you for: **Volume Definition** (*n*), where *n* is the number corresponding to the disk volume definition you want to remove.

For example, suppose you defined three volumes: first, C; then, E; then, D. Essbase considers D the third volume - definition number 3.

Note: If you delete an application or database, Essbase does not remove the directory containing the application or database on a disk volume. The computer's operating system still shows the folder and file labels on the disk. However, you can reuse the same name of the application or database that you had removed on the disk volume.

- **26. Cache Memory Locking**—Enable or disable Cache Memory Locking. When enabled, this setting locks the memory used for the index cache, data file cache, and data cache into physical memory, improving database performance.

Values: Y/N Default: No

- **27. Data File Cache Size**—

Specify the size, in bytes, for the Data File Cache. Minimum: 8388608 bytes. Default: 33554432 bytes. Recommended: Combined size of all ESS*.PAG files if possible; as large as possible otherwise.

- **99. All Items**—Prompts for each option in turn. Use only in interactive mode.

Example

The following example enables Committed access and Pre-image access, and specifies indefinite wait time:

```
SETDBSTATEITEM 18 "JTEMP" "JTEMPDB" "1" "Y" "1";
```

The following example allocates up to 4 GB on Volume E, sets a maximum file size of 1 GB, and specifies that data files should be stored only on E:

```
SETDBSTATEITEM 23 "SAMPLE" "BASIC" "1" "E" "4G" "2" "1G"
```

The following examples set the data cache value to 45000000 bytes. In the first example, the SELECT command was used to select the application and database. In the second example, the application and database are specified in the SETDBSTATEITEM command line instead.

Example 1 (SETDBSTATEITEM)

```
LOGIN "machinename" "admin" "password";
SELECT "Sample" "Basic";
SETDBSTATEITEM 5 45000000;
LOGOUT;
EXIT;
```

Example 2 (SETDBSTATEITEM)

```
LOGIN "machinename" "admin" "password";
SETDBSTATEITEM 5 sample basic 45000000;
LOGOUT;
EXIT;
```

See Also

- [SELECT](#)

SETDEFAULTCALC

Sets a calculation string as the default database calculation.

Syntax

```
SETDEFAULTCALC calcString
```

Parameter Description

calcString Calculation string to set.

Notes

- Place the default database calculation within quotation marks.
- Calculation strings require a terminating semicolon.

Example

```
SETDEFAULTCALC "CALC ALL;";
```

See Also

- [SETDEFAULTCALCFILE](#)

SETDEFAULTCALCFILE

Sets a calculation object as the default database calculation.

Syntax

```
SETDEFAULTCALCFILE calcobjName
```

Parameter Description

calcobjName Calculation object to set. Give full path name if this object is not in the CLIENT directory.

Example

```
SETDEFAULT "actbud";
```

See Also

- [SETDEFAULTCALC](#)

SETLOGIN

Sets the active login to a particular instance.

Syntax

```
SETLOGIN sesNo
```

Parameter Description

sesNo Login instance session number. Values:

- `prev` - Previous number
- `next` - Next session number
- `sessionNo` - Integer representing session

Notes

This command sets the active login to the instance represented by previous, next, or a session number. To get session numbers, use the LISTLOGINS command.

Example

To set the ESSCMD session to login the previous login instance:

```
SETLOGIN PREV;
```

To set the ESSCMD session to login the next login instance:

```
SETLOGIN NEXT
```

To set the ESSCMD session to login instance number 2:

```
SETLOGIN 2
```

The following interactive example uses LISTLOGINS to get information needed for a subsequent SETLOGIN command. Commands typed by the user are shown in **bold**.

```
localhost:::system[1]->listlogins  
ListLogins:
```

```
There are 2 Active Login Sessions.  
  Login Session 1 -- localhost  system  
  Login Session 2 -- localhost  EWhite
```

```
localhost:::system[1]->setlogin 2  
SetLogin:
```

```
Switch to Login Session 2 -- localhost  EWhite
```

See Also

- [LISTLOGINS](#)

SETMSGLEVEL

Defines the level of messages seen in the interactive ESSCMD shell.

Syntax

```
SETMSGLEVEL level
```

Parameter Description

level	Level setting for messages. Values: 1 - Make no changes 2 - Display all information messages 3 - Display only warning messages 4 - Display only error messages 5 - Display no messages
-------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Notes

The SETMSGLEVEL command defines the level of messages seen in the interactive ESSCMD shell. To set the level of messages seen in an ESSCMD output file, use the [OUTPUT](#) command.

Example

```
SETMSGLEVEL 3;
```

See Also

- [OUTPUT](#)
- [“AGENTLOGMESSAGELEVEL” on page 386](#)
- [“Set Message Level” on page 835](#)

SETPASSWORD

Assigns a new password to an existing user.

Syntax

```
SETPASSWORD userName newPassword
```

Parameter Description

userName	Name of the existing user.
newPassword	New password for the user.

Example

```
SETPASSWORD "D_ROSETTI" "INFERNO";
```

SHUTDOWNSERVER

Shuts down the Essbase Server from the terminal running the current ESSCMD session. You must have Administrator permission to use this command.

Syntax

```
SHUTDOWNSERVER servername username password
```

Parameter Description

servername Host name associated with the Essbase Server you want to shut down.

username Your user name.

password Your password.

Notes

If you do not specify the parameters on SHUTDOWNSERVER, ESSCMD prompts you for them.

Example

To shut down the Essbase Server named Poplar:

```
SHUTDOWNSERVER "poplar" "mildred" "password";
```

To have Essbase prompt you for your user name and password:

```
SHUTDOWNSERVER "Poplar";
```

SLEEP

Pauses an ESSCMD script.

Syntax

```
SLEEP "seconds"
```

Parameter Description

seconds Number of seconds for the batch file execution to sleep.

Notes

Pauses an ESSCMD script. Pausing an ESSCMD script allows other commands to finish execution and cleanup.

Example

```
SLEEP "10";
```

UNLOADALIAS

Deletes the specified alias table.

Syntax

```
UNLOADALIAS aliasName
```

Parameter Description

aliasName Name of the alias table to unload.

Example

Assume that `flavors` is an alias table mapping the following flavor names to the numerically-named children of Product:

```
"400-10"      Guava
"400-20"      Tangerine
"400-30"      Mango
```

These flavors are discontinued. To delete the alias table called `flavors`, first select the application and database, and then enter the following:

```
UNLOADALIAS "flavors";
```

See Also

- [LISTALIASES](#)
- [DISPLAYALIAS](#)
- [SETALIAS](#)

UNLOADAPP

Unloads an application from memory.

Syntax

```
UNLOADAPP appName
```

Parameter Description

`appName` Name of the application to unload.

Notes

All databases within the application are unloaded.

Example

```
UNLOADAPP "SAMPLE";
```

UNLOADDB

Unloads a database from memory.

Syntax

```
UNLOADDB appName dbName
```

Parameter Description

`appName` Name of the application in which the database resides.

`dbName` Name of the database to unload.

Example

```
UNLOADDB "SAMPLE" "BASIC";
```

UNLOCKOBJECT

Unlocks an object that is locked by another user or process.

Syntax

```
UNLOCKOBJECT objType sourceApp sourceDb sourceObj
```

Parameter Description

objType Type of object to list. Values:

- 1 - Outline object.
- 2 - Calculation script.
- 3 - Report script.
- 4 - Rules object.
- 5 - Alias table (not available).
- 6 - Structure file (not available).
- 7 - Backup file (not available).
- 8 - Worksheet of any type (not available).
- 9 - Text object.
- 10 - Partition.
- 11 - Linked Reporting Object (stored).
- 12 - Selection.
- 13 - Wizard.
- 14 - EQD.

sourceApp Name of the application that includes object.

sourceDb Name of the database that includes object.

sourceObj Name of the existing object to unlock.

Notes

- Values 5 through 8 for the *objType* parameter represent objects that cannot be locked.
- Two values for the *objType* parameter, 6 and 7, are retained only for backward compatibility with Release 2.0.
- Option 11, Linked Reporting Object, unlocks stored LROs only; that is, files with the .LRO extension. It does not unlock URLs, cell notes, or linked partitions.

Example

```
UNLOCKOBJECT 1 "FINANC95" "SALES95" "ARCHIVE";
```

UPDATE

Loads a single data record into the selected database.

Syntax

```
UPDATE dataString
```

Parameter Description

dataString A single data record.

Example

```
UPDATE "Jan Sales '100-10' Florida Actual 220";
```

See Also

- [UPDATEFILE](#)

UPDATEBAKFILE

Compares the security backup file, *essbase_timestamp.bak*, to the security file, *essbase.sec*, at any time, and if needed, triggers an update. The backup file is updated only if a difference exists between the security file, *essbase.sec*, and the security backup file, *essbase_timestamp.bak*.

Syntax

```
UPDATEBAKFILE
```

Example

```
UPDATEBAKFILE
```

UPDATEFILE

Loads data, unlocks blocks, and verifies a data file.

Syntax

```
UPDATEFILE location fileName update
```

Parameter Description

<i>location</i>	Location of the data file. Values: <ul style="list-style-type: none">1 - Local/client-based rules file2 - Remote/server data object3 - File. Use option 3 if you want to specify the full path name. Otherwise, the file is assumed to be in the <appName>/<DBNAME> directory.
<i>fileName</i>	Name of the file to load.
<i>update</i>	Update action. Values: <ul style="list-style-type: none">1 - Load data2 - Unlock data blocks3 - Verify data

Example

```
UPDATEFILE 2 "DATA" 1;
```

See Also

- [LOADDATA](#)

UPDATEVARIABLE

Updates the variable value that corresponds to the specified substitution variable.

Syntax

```
UPDATEVARIABLE variableName [serverName [appName [dbName]]] value
```

Parameter	Description
<code>variableName</code>	The name of the existing substitution variable.
<code>serverName</code>	Optional. Host name of the Essbase Server to which the variable is applied.
<code>appName</code>	Optional. Name of the application to which the variable is applied. If <code>appName</code> is not used, in a script, empty quotes must be used to take its place. (" ")
<code>dbName</code>	Optional. Name of the database to which the variable is applied. If <code>dbName</code> is not used, in a script, empty quotes must be used to take its place. (" ")
<code>value</code>	The new string value that corresponds to the substitution variable. The name must be alphanumeric, and can be a maximum of 255 characters. You can have a null value, but do not use a leading & character in the value.

Example

The following command in an ESSCMD script updates a substitution variable named `CurQtr` to have a value of `Qtr2`.

```
UPDATEVARIABLE "CurQtr" "Bamboo" "Sample" "Basic" "Qtr2";
```

The following ESSCMD script updates a substitution variable named `CurQtr` to have a value of `Qtr3`. Application and database input is left blank because the variable is system-wide; however, the empty quotation marks are still required as placeholders.

```
login "Aspen" "fiona" "sunflower";  
UPDATEVARIABLE "CurQtr" "aspen" "" "" "Qtr3";
```

Another script that updates an Essbase Server substitution variable:

```
OUTPUT 1 "subvar_serv.log";  
LOGIN "localhost" "system" "password";  
UPDATEVARIABLE "GlobalVar" "" "" "" "Myserver";  
exit;
```

Script that updates an application substitution variable:

```
OUTPUT 1 "subvar_app.log";  
LOGIN "localhost" "system" "password";
```

```
UPDATEVARIABLE "AppVar" "localhost" "Sample" "" "MyApp";
exit;
```

Script that updates a database substitution variable:

```
OUTPUT 1 "subvar_db.log";
LOGIN "localhost" "system" "password";
UPDATEVARIABLE "DBVar" "localhost" "Sample" "Basic" "MyDB";
exit;
```

See Also

- [LISTVARIABLES](#)
- [UPDATEVARIABLE](#)

VALIDATE

Checks the database for data and structural integrity. You must select a database before issuing this command.

VALIDATE checks the following information:

- Verifies data integrity in each block. Reading from top to bottom, it checks blocks, sections, block type, and block length. The command checks for validity in floating-point numbers. This command writes information about bad blocks to the log file.
- Automatically compares every index key in the index page with the index key in the corresponding data block and checks other header information in the block. If it encounters a mismatch, VALIDATE displays an error message and continues processing until it checks the entire database.
- Compares the data block key in the index page with the data block key in the corresponding data block. Keys out of order indicate corruption.
- Verifies the structural integrity of the index free space information in the index.
- Verifies the structural integrity of the LRO catalog.

If this command finds integrity errors, it writes validation process error messages to a text-format log file. The default location for the specified file is in the application\database directory. For example: ESSBASE\APP\app\db\VALIDATE.LST.

Syntax

```
VALIDATE errorLogFile
```

Parameter	Description
-----------	-------------

errorLogFile	Name and optional path of destination file for error messages. If no path is specified, the specified list file is stored in the current application\database directory.
--------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Notes

- You can also use the VALIDATE command to clear an internal file, *database_name*.OCL, when it grows too large. *database_name*.OCL is a file used for

incremental restructuring. VALIDATE causes Essbase to restructure any blocks whose restructure was deferred, and clears the file.

- Before issuing the VALIDATE command, we recommend placing the database in read-only mode, using the ESSCMD BEGINARCHIVE or the MaxL statement **alter database DBS-NAME begin archive to file FILE-NAME;**

Example

```
VALIDATE VALERROR.TXT;
```

VALIDATEPARTITIONDEFFILE

Validates shared partition definitions.

Syntax

```
VALIDATEPARTITIONDEFFILE
```

Notes

This command validates the specified partition definition identified in the partition mapping definition .DDB file. During validation, Essbase checks the .DDB file to ensure that:

- The area definition is valid (contains no syntax errors).
- The specified data source members are valid members and map to valid members in the data target.
- All connection information is correct (host names, database names, application names, user names, and password information).
- For linked partitions, the specified default user name and password are correct.
- For replicated and transparent partitions:
 - A replication target does not overlap with replication target.
 - A replication target does not overlap with transparent target.
 - A transparent target does not overlap with transparent target.
 - A replication source does not overlap with transparent target.
 - The cell count for the partition is the same on the data source and the data target.

You must issue the VALIDATEPARTITIONDEFFILE command for both the data source and the data target .DDB files. You need to log in to each database and issue the command separately for each portion of the partition definition.

For more information, see the *Oracle Essbase Database Administrator's Guide*.

Example

```
VALIDATEPARTITIONDEFFILE
```

In This Chapter

Overview of MaxL and MDX	623
How to Read MaxL Railroad Diagrams	624
MaxL Data Definition Language (DDL)	626
MaxL Statements	626
MaxL Definitions	767
MaxL Shell Commands	818
MaxL Perl Module	841
ESSCMD Script Conversion	848
Reserved Words List	855
MaxL Statements (Aggregate Storage)	864
Outline Paging Dimension Statistics	904
Aggregate Storage Runtime Statistics	905
MaxL Statements for Data Mining	907
MaxL Use Cases	920

Overview of MaxL and MDX

MaxL is the multi-dimensional database access language for Essbase. MaxL is a practical, expressive interface for administering and querying the Essbase system. With the MaxL language, you use statements to make requests. MaxL statements usually begin with a verb, and read like English sentences.

Beginning with Release 7.0, MaxL has two functional domains:

- MaxL DDL is the [data-definition language](#) for Essbase.

Data definition means structural control of a database system. This includes operations like creation, deletion, and updating of users, applications, databases, and database objects.

Therefore, statements in MaxL DDL include verbs like CREATE, ALTER, DROP, GRANT, and DISPLAY.

- MDX is the [data-manipulation](#) language for Essbase.

Data manipulation means access to the actual data within a database system. MDX provides the ability to perform advanced data extraction and querying by means of statements that typically include the verb SELECT. The equivalent conceptual tool would be Report Writer.

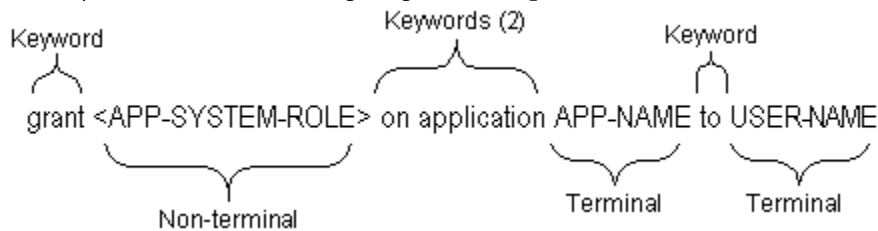
How to Read MaxL Railroad Diagrams

The MaxL [grammar](#) is illustrated using a railroad syntax notation. The railroad diagrams illustrate all the valid (grammatically correct) statements that can be parsed by MaxL.

- “Anatomy of MaxL Statements” on page 624
- “Railroad Diagram Symbols” on page 624
- “Sample Railroad Diagram” on page 625

Anatomy of MaxL Statements

- A [keyword](#) (see `grant`, represented in plain, lower-case font, is a unit of MaxL grammar. Keywords must be entered literally and in the correct order in MaxL statements. See the examples of keywords in the following diagram excerpt:



- A [terminal](#), represented in upper-case without brackets, is replaced by values in the appropriate format as defined in the [Terminals](#) table. In the above diagram, `APP-NAME` and `USER-NAME` are examples of terminals. Each would need to be replaced with a valid name; for example, `sample` or `user1`.

Keywords cannot be used as terminals, unless enclosed in single quotation marks. For example, to delete a user named `user`, the statement `drop user user;` would return an error, but `drop user 'user';` would work.



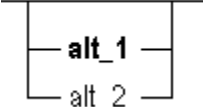
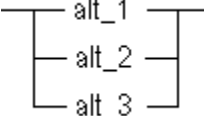
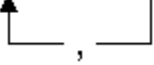
- A non-terminal, represented in upper-case with angle brackets `<>`, is defined in an additional diagram, usually below the main diagram.

Keywords and variables on the main line are required; optional grammar is recessed. A vertical stack of words represents alternatives. Bold words indicate defaults when no word is chosen.

Railroad Diagram Symbols

The following table describes the meaning of symbols used in railroad diagrams.

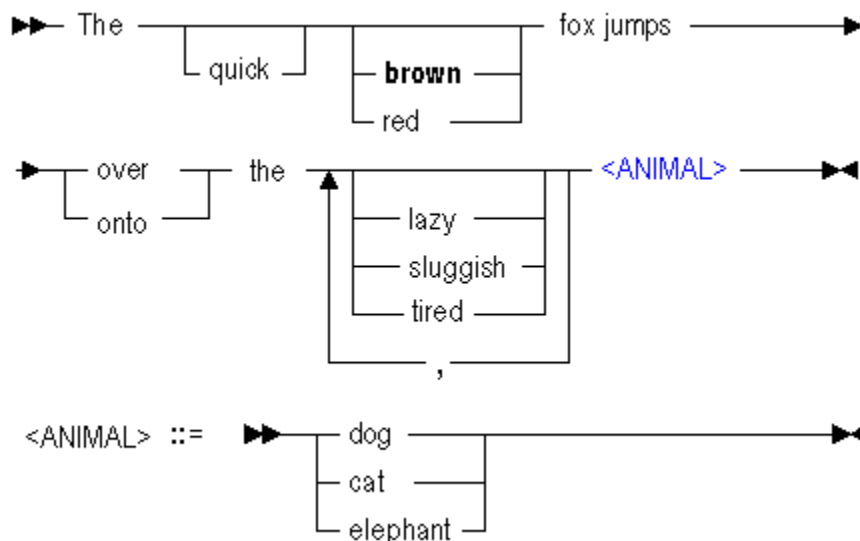
Symbol	Definition
	Statement begins here.
	Statement continues on next line.

Symbol	Definition
	Statement is continued from previous line.
	Statement ends here.
	Alternatives: optionally select one keyword. Boldface indicates default if no selection is made.
	Alternatives: selection of one keyword is required.
	A comma-separated list of any length is permitted.
TERMINAL-NAME	Word is not further defined. Replace with value of format shown in the Terminals table.
<NON-TERMINAL>	Word used in statement is further defined.
<NON-TERMINAL> ::=	Non-terminal used in statements is defined here.

Sample Railroad Diagram

The following diagram illustrates a variant grammar that parses the following English sentence:

"The quick brown fox jumps over the lazy dog."



Valid sentences parsable by this grammar:

- The fox jumps over the dog. Bold letters indicate a default value when no option is entered; therefore, entry of this statement would be interpreted as *The brown fox jumps over the dog*.
- The quick brown fox jumps over the dog.
- The red fox jumps over the lazy cat.
- The quick brown fox jumps onto the tired elephant.

MaxL Data Definition Language (DDL)

MaxL DDL is the database definition language for Essbase. MaxL DDL is a practical, expressive interface for administering Essbase. With the MaxL DDL language, you use statements to make requests. MaxL DDL statements begin with a verb and read like English sentences.

In order for Essbase Server to receive MaxL DDL statements, you must pass the statements to Essbase Server. To pass statements, you use either MaxL Shell (essmsh), MaxL Script Editor in the Administration Services Console, or the MaxL Perl Module.

It is recommended that you proceed in the following order:

1. Start Essbase Server.
2. Invoke [MaxL Shell](#) or MaxL Script Editor and log in to Essbase Server.

Note: For information about MaxL Script Editor, see the *Oracle Essbase Administration Services Online Help*.

3. Create [statements](#) for data access and system administration.
4. Learn about syntax, numbers, permissions, and names in the MaxL language (see “[MaxL Definitions](#)” on page 767).
5. Learn about using [Perl](#) to issue MaxL statements.

MaxL Statements

The MaxL data-definition language has its own grammar that you use to create statements. In this document, the syntax for the MaxL DDL is illustrated using railroad diagrams.

The MaxL grammar is case-insensitive. Semicolon statement-terminators are required when using the MaxL Shell. However, do not use semicolons at the end of statements passed using Perl functions.

Key words of the MaxL grammar are represented in this document in lower-case. Terminals, represented in upper-case, are to be replaced by the appropriate names, numbers, privileges, or strings. For more information about components of MaxL statements, see “[MaxL Definitions](#)” on page 767.

Topics covered in this section:

- [“Performance Statistics in MaxL” on page 627](#)
- [“Listed By Verbs” on page 633](#)
- [“Object” on page 641](#)
- [“MaxL Statement Reference” on page 642](#)

Performance Statistics in MaxL

`Query database` returns medium and long performance statistics for the database and application. The statistics appear as tables in the MaxL output. To gather performance statistics, you must first enable statistics gathering using `alter database <db-name> set performance statistics enabled`. You also use `alter database` to return to zero the statistical *persistence* (length) and *scope* (granularity).

Collecting and analyzing performance statistics can help you understand whether the databases are in good running condition or could use modifications to improve performance.

Topics related to performance statistics:

- [“The Essbase Performance Statistics Tables” on page 627](#)
- [“MaxL Script Example” on page 632](#)

The Essbase Performance Statistics Tables

The Essbase system gathers a variety of statistics regarding the performance of the system and the connected applications. The output of `query database` can vary depending on what the system has just done, how long statistics have been gathered and the persistence of the gathered statistics. The tables give information on a typical set of statistics. It can be very helpful to compare two sets of statistics gathered at similar points in the server's operation, such as after two comparable updates or after two restructure operations. Statistics should be gathered at intervals and compared to each other to identify differences. Compare the statistics gathered before and after any changes to the system and if the system performance changes.

Note: Depending on the calculations you choose to perform, if any, some tables may or may not be displayed in your output log.

Performance statistics for which tables are available:

- [“Kernel Input/Output Statistics” on page 628](#)
- [“Kernel Cache Statistics” on page 628](#)
- [“Cache End-Transaction Statistics” on page 629](#)
- [“Database Synchronous Input/Output Statistics” on page 629](#)
- [“Database Asynchronous Input/Output Statistics” on page 630](#)
- [“Dynamic Calc Cache Statistics” on page 631](#)

Kernel Input/Output Statistics

The **Kernel I/O Statistics** table summarizes input/output for the entire application. There is one kernel I/O table per application.

Persistence/Scope of this table: **med/server**

Kernel I/O	Read (OS reads from disk)	Write (OS writes to disk)
# Index	I/O Number of reads that occurred through the index cache.	Number of writes that occurred through the index cache.
# Data I/O	Number of reads that occurred through the data cache.	Number of writes that occurred through the data cache.
# Fground I/O	Number of data reads that occurred in the foreground (while a process waited for data to be read).	Number of data writes that occurred in the foreground (while a process waited for data to be written).
# Index bytes	Number of bytes read from .IND files.	Number of bytes written to .IND files.
# Data bytes	Number of bytes read from .PAG files.	Number of bytes written to .PAG files.
Av byte/dat I/O	Average byte size of data reads. A high number is preferable.	Average byte size of data writes. A high number is preferable.

Kernel Cache Statistics

The **Kernel Cache Statistics** table assists in sizing database caches. Make caches only as large as necessary for optimum performance. Note that cache sizes are listed in order of importance: index, data file, data.

- The index cache is a buffer in memory that holds index pages.
- The data file cache is a physical data cache layer designed to hold compressed data blocks.
- The data cache is a buffer in memory that holds data pages.

The Kernel Cache Statistics table assists you in determining how to size Essbase caches. The Essbase kernel uses these caches to manage memory. As a rule, data that is useful to processes should be kept in memory rather than on a disk. Replacements occur when something needed for a process is moved from disk to cache and something in the cache is thrown away to make room for it.

Use this table to help you decide how to size your caches. Make the caches as small as possible; however, if replacements for a cache are greater than 0, the cache may be too small. Appropriate sizing of the Index cache is the most important for optimal performance; appropriate sizing of the Data cache is the least important.

Persistence/Scope of this table: **long/db**

Kernel Cache Statistic	Description
# Blocks	Number of blocks actually in the Index cache, Data file cache, and Data cache. The block size multiplied by the number of blocks equals the amount of cache memory being used. Compare this figure to the block estimation you initially used to size your database (see the <i>Oracle Essbase Database Administrator's Guide</i>).
# Replacements	Number of replacements per cache. Replacements occur when data moves from disk to cache and something in the cache is deleted to make room. If the number of replacements is low or zero, the cache might be set too large.
# Dirty repl	Number of dirty replacements per cache. A dirty replacement is one that requires a write to the disk before cache memory can be reused by a process. The data needed for the process is "dirty" because it was modified in memory but not saved to the disk. Dirty replacements are inefficient and expensive. They indicate that a cache might be too small.
Log blk xfer in	Number of logical blocks transferred to the Data file cache and Data cache (this measurement is not applicable for the Index cache.) If you are changing cache sizes, it may be instructive to study this statistic and note changes in data traffic.

Cache End-Transaction Statistics

The **Cache End-Transaction Statistics** table measures DBWriter efficiency. DBWriter is an asynchronous (or no-wait) Essbase thread, which searches the cache finding information that needs to be written to a disk.

The Cache End-Transaction Statistics table shows the cleanup state at the end of a transaction. These statistics are designed to measure DBWriter efficiency. DBWriter is an asynchronous (or no-wait) thread, which searches the cache and finds information that needs to be written to a disk. Because the DBWriter only operates during idle times, measuring the DBWriter activity can give an idea of the amount of idle time. This number should be high, indicating that the DBWriter had enough idle time to support the database effectively. Keep these statistics available for diagnostic purposes, in case you need to call technical support.

Persistence/Scope of this table: **med/db**

Database Synchronous Input/Output Statistics

The **Database Synchronous I/O** table tracks synchronous input/output. Synchronous means that the thread or program waits for the I/O to finish before proceeding. The **Tave (us)** column shows the bandwidth (bytes/Ttotal).

Persistence/Scope of this table: **med/db**

DataBase Synchron I/O	Count	Bytes	Ttotal (ms)	Tave (ms)
Index Read An occurrence of the OS reading index information from a .IND file on the disk.	Number of times the OS went to the disk to read a .IND file.	Number of bytes the OS read from .IND files.	Total amount of time the OS took to complete index reads.	Average amount of time the OS took to complete one index read. This equals Ttotal (ms)/Count.
Index Write An occurrence of the OS writing index information to a .IND file.	Number of times the OS wrote information to a .IND file.	Number of bytes the OS wrote to .IND files.	Total amount of time the OS took to complete index writes.	Average amount of time the OS took to complete one index write. This equals Ttotal (ms)/Count.
Data Read An occurrence of the OS reading information from a .PAG file on the disk.	Number of times the OS went to the disk to read to a .PAG file.	Number of bytes the OS read from .PAG files.	Total amount of time the OS took to complete data reads.	Average amount of time the OS took to complete one data read. This equals Ttotal (ms)/Count.
Data Write An occurrence of the OS writing data to a .PAG file.	Number of times the OS wrote information to a .PAG file.	Number of bytes the OS wrote to .PAG files.	Total amount of time the OS took to complete data writes.	Average amount of time the OS took to complete one data write. This equals Ttotal (ms)/Count.

Note: Bandwidth = bytes/Ttotal. Average bandwidth= bytes/Tave.

Database Asynchronous Input/Output Statistics

The **Database Asynchronous I/O** table tracks asynchronous input/output. Asynchronous means no-wait: the I/O happens at an unknown time, while the program does other things. The effective bandwidth for the application is determined by bytes/Twait.

Persistence/Scope of this table: **med/db**

DataBase Asynch I/O	Count	Bytes	Ttotal (ms)	Tave (ms)	Twait (ms)
Index Read An occurrence of the OS reading index information from a .IND file on the disk.	Number of times the OS went to the disk to read a .IND file.	Number of bytes the OS read from .IND files.	Time elapsed between request for an index read, and verification of its completion.	Average time elapsed between requests for index reads, and verification of their completion.	Wait time if the OS had not completed index reads at the time polled.
Index Write An occurrence of the OS writing index information to a .IND file.	Number of times the OS wrote information to a .IND file.	Number of bytes the OS wrote to .IND files.	Time elapsed between request for an index write, and verification of its completion.	Average time elapsed between requests for index writes and verification of their completion.	Wait time if the OS had not completed index writes at the time polled.
Data Read An occurrence of the OS reading information from a .PAG file on the disk.	Number of times the OS went to the disk to read to a .PAG file.	Number of bytes the OS read from .PAG files.	Time elapsed between request for a data read, and verification of its completion.	Average time elapsed between requests for data reads, and verification of their completion.	Wait time if the OS had not completed data reads at the time polled.
Data Write An occurrence of the OS writing data to a .PAG file.	Number of times the OS wrote information to a .PAG file.	Number of bytes the OS wrote to .PAG files.	Time elapsed between request for a data write, and verification of its completion.	Average time elapsed between requests for data writes and verification of their completion.	Wait time if the OS had not completed data writes at the time polled.

Note: (1) Because asynchronous I/O is ideally no-wait, and happens at an unknown time, you cannot determine how long reads and writes actually took to complete. (2) You cannot determine the bandwidth (bytes per microsecond). Effective bandwidth, as seen by the application, is determined by bytes/Twait.

Dynamic Calc Cache Statistics

The **Dynamic Calc Cache** table shows where blocks that are expanded to contain calculated members (BigBlks) are calculated: in dynamic calculator cache (DCC), or in regular memory (nonDCC). By viewing the total number of big blocks allocated versus the maximum number of big blocks held simultaneously, and by analyzing block wait statistics, you can determine the efficiency of your dynamic calc cache configuration settings. For more information, refer to the [“DYNCALCCACHEMAXSIZE” on page 433](#) setting in the `essbase.cfg` documentation.

Dynamic Calc Cache Statistic	Description
BigBlks Allocated	The number of big block allocations that have been requested, so far, irrespective of where the system got the memory (DC cache or regular). For three queries Q1, Q2, and Q3 executed, requiring 25, 35, and 10 big blocks, respectively, BigBlks Allocated would be 70. This does not mean that Q1 needed all 25 blocks at the same time. It may have used some blocks for a while, then released some of them, and so on, until the query finished and released all remaining blocks (returned to DC cache or regular memory).
Max BigBlks Held	The maximum number of big blocks simultaneously held, so far. For each query Qi executed so far, there will be a number Ni, which gives the maximum number of big blocks that the query needed to have at the same time (includes both DCC and regular memory blocks). MaxBigBlksHeld under the Total column is the maximum over all values of Ni. The values under the DCC and non-DCC columns are similar except that they restrict themselves to the maximum blocks held in the respective portions of memory.
DCC Blks Waited	The number of dynamic calculator blocks that the system had to wait for.
DCC Blks Timeout	The number of times that the “ DYNCALCCACHEBLKTIMEOUT ” on page 430 configuration setting was exceeded.
DCC Max ThdQLen	If the configuration setting, “ DYNCALCCACHEWAITFORBLK ” on page 436 is TRUE, it is possible for queries (really, the threads executing them) to sit in a queue, waiting for DC cache memory to be freed by other threads currently using the memory. DCC MaxThdQLen tells how long this queue ever got (maximum number of threads simultaneously waiting), giving a sense of how critical the dynamic calculator cache became as a resource.

MaxL Script Example

The following MaxL script creates an output file of performance statistics tables.

```

/* to execute:
   essmsh scriptname username password
*/
login $1 $2;
spool on to 'c:\mxlouts\pstatsouts.txt';
alter database sample.basic set performance statistics enabled;
execute calculation
  'SET MSG ERROR;
   CALC ALL; '
on Sample.basic;
alter database sample.basic set performance statistics mode to medium persistence server
scope;
query database sample.basic get performance statistics kernel_io table;
alter database sample.basic set performance statistics mode to long persistence database
scope;
query database sample.basic get performance statistics kernel_cache table;
alter database sample.basic set performance statistics mode to medium persistence
database scope;
query database sample.basic get performance statistics end_transaction table;
query database sample.basic get performance statistics database_synch table;
query database sample.basic get performance statistics database_asynch table;
spool off;
logout;

```


Listed By Verbs

alter
create
deploy
display
drop
execute
export
grant
import
query
refresh

Alter

application
database
drillthrough
filter
group
object
partition
session
system
tablespace
trigger
user

Create

application
calculation
database
drillthrough
filter

function
group
location alias
macro
outline
partition
trigger
user
(data mining) algorithm
(data mining) model
(data mining) task template

Deploy

deploy

Display

application
calculation
database
disk volume
drillthrough
filter
filter row
function
group
location alias
lock
macro
object
partition
privilege
session
system

tablespace
trigger
trigger spool
user
variable
(data mining) algorithm
(data mining) model
(data mining) session
(data mining) task template

Drop

application
calculation
database
drillthrough
filter
function
group
location alias
lock
macro
object
partition
trigger
trigger spool
user
(data mining) algorithm
(data mining) model
(data mining) result
(data mining) task template

Execute

aggregate process

[aggregate selection](#)

[aggregate build](#)

[allocation](#)

[calculation](#)

[custom calculation](#) (aggregate storage)

Export

[data](#)

[LRO](#)

[outline](#)

[security_file](#)

Grant

[Grant](#)

Import

[data](#)

[dimensions](#)

[lro](#)

Query

[database](#)

[database backup archive file](#)

[application](#) (for aggregate storage only)

Refresh

[custom definitions](#)

[outline](#)

[replicated partition](#)

Listed by Objects

[aggregate_build](#)

[aggregate_process](#)

aggregate_selection
application
archive_file
calculation
custom definitions
data
database
data mining objects
dimensions
disk volume
drillthrough
filter
function
group
location alias
lock
lro
macro
object
outline
partition
privilege
security_file
session
system
tablespace
trigger
trigger spool
user
variable

Aggregate Build

execute aggregate build

Aggregate Process

execute aggregate process

Aggregate Selection

execute aggregate selection

Allocation

execute allocation

Application

alter

create

display

drop

query (for aggregate storage only)

Archive_file

query

Calculation

create

display

drop

execute

execute custom (aggregate storage)

Custom Definitions

create function

create macro

display function

display macro

drop function

drop macro

refresh custom definitions

Data

export

import

Database

alter

create

display

drop

query

Dimensions

import

Disk Volume

alter database (to add, drop, and set)

display disk volume

Drillthrough

alter

create

display

drop

Filter

alter filter

create filter

display filter

display filter row

drop filter

Function

create
display
drop
refresh

Group

alter
alter user (to add or remove group members)
create
display
drop

Location Alias

create
display
drop

Lock

display
drop

LRO

export
import

Macro

create
display
drop
refresh

Object

alter
display
drop

Outline

create
refresh
see also “Dimensions” on page 639

Partition

alter
create
display
drop
refresh replicated
refresh outline for outline synchronization

Privilege

display
grant

Security File

Export Security File

Session

alter
display
alter system to stop a session

System

alter
display

Tablespace

[alter](#)
[display](#)

Trigger

[alter](#)
[create or replace](#)
[display](#)
[drop](#)

Trigger Spool

[display](#)
[drop](#)

User

[alter](#)
[create](#)
[display](#)
[drop](#)
[grant](#) to assign permissions

Variable

[display variable](#)
To add, drop, or set substitution variables:
[alter application](#)
[alter database](#)
[alter system](#)

MaxL Statement Reference

Consult the Contents pane for an alphabetical list of MaxL statements.

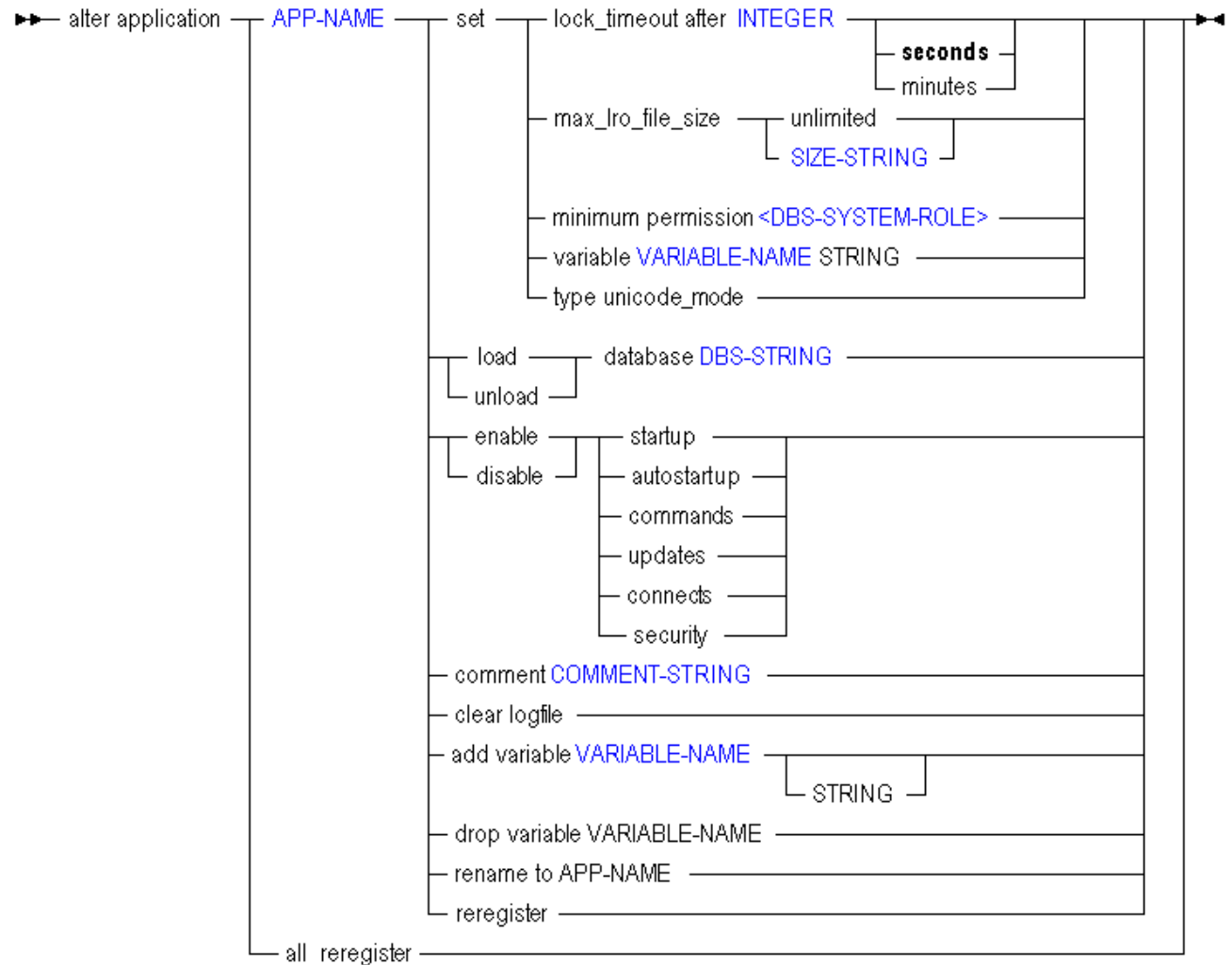
Alter Application

[Click here for aggregate storage version](#)

Change application-wide settings.

Permission required: Application Manager.

Syntax



Use **alter application** to change the following application-wide settings:

Keyword	Description
set lock_timeout	Change the maximum time interval that locks on data blocks can be held by Spreadsheet Add-in users. When a client data-block lock is held for more than the time out interval, Essbase removes the lock and the transaction is rolled back. The default interval is 60 minutes. This setting affects all databases in the application.
set max_lro_file_size	Specify a maximum file size for Linked Reporting Objects (LRO) attachments. There is no default. There is no minimum or maximum value, excepting limitations imposed by your system resources.
set minimum permission	Grant all users a minimum level of permission to all databases in the application. Users with higher permissions than this minimum are not affected.
set variable	Assign a string value to an existing substitution-variable name. If the variable does not exist, first create it using add variable . Substitution variables may be referenced by calculations in the application.

Keyword	Description
set type unicode_mode	Migrate an application to Unicode mode. Migration to Unicode mode cannot be reversed.
load database	Start (by loading into memory) an idle database. The statement will fail if you do not have at least read privilege for the database.
unload database	Stop (by unloading from memory) an active database. The statement will fail if you do not have at least read privilege for the database.
enable startup	Permit all users to load (start) the application. This only applies to users who have at least read privilege for the application. Startup is enabled by default.
disable startup	Prevent all users from loading (starting) the application. Startup is enabled by default.
enable autostartup	Start the application automatically when Essbase Server starts. By default, autostartup is disabled.
disable autostartup	Do not start the application automatically when Essbase Server starts. By default, autostartup is disabled.
enable commands	Allow all users with sufficient permissions to make requests to databases in the application. Use to reverse the effect of disable commands . The disable commands setting remains in effect only for the duration of your session. By default, commands are enabled.
disable commands	Prevent all requests to databases in the application, including non-data-specific requests, such as viewing database information or changing database settings. All users are affected, including other administrators. Administrators are affected by this setting as a safety mechanism to prevent accidental updates to databases during maintenance operations. This setting remains in effect only for the duration of your session. The setting takes effect immediately, and affects users who are currently logged in, as well as users who log in later during your session.
	<hr/> <p>Caution! If performing maintenance operations that require disabling commands, you must make those maintenance operations within the same session and the same script as the one in which commands were disabled.</p> <hr/>
	By default, commands are enabled.
enable updates	Allow all users with sufficient permissions to make requests to databases in the application. Use to reverse the effect of disable updates . Disabling updates remains in effect only for the duration of your session. By default, updates are enabled.
disable updates	Prevent all users from making requests to databases in the application. Use before performing update and maintenance operations. The disable updates setting remains in effect only for the duration of your session.
	<hr/> <p>Caution! If performing maintenance operations that require updates to be disabled, you must make those maintenance operations within the same session and the same script as the one in which updates were disabled. By default, updates are enabled.</p> <hr/>
enable connects	Allow all users with sufficient permissions to make connections to databases in the application. Use to reverse the effect of disable connects . By default, connections are enabled.

Keyword	Description
disable connects	Prevent any user with a permission lower than Application Managers from making connections to the databases that require the databases to be started. This includes starting the databases or performing the ESSCMD SELECT command on the databases. Database connections remain disabled for all databases in the application, until the application setting is re-enabled by the administrator. By default, connections are enabled.
enable security	When security is disabled, Essbase ignores all security settings in the application and treats all users as Application Managers. By default, security is enabled.
disable security	When security is disabled, Essbase ignores all security settings in the application and treats all users as Application Managers. By default, security is enabled.
comment	Enter an application description (optional). The description can contain up to 80 characters.
clear logfile	Delete the application log located in the application directory. A new log is created for entries recording subsequent application activity.
add variable	Create an application-level substitution variable by name, and optionally assign a string value for the variable to represent. You can assign or change the value later using set variable . A substitution variable acts as a global placeholder for information that changes regularly. Substitution variables may be referenced by calculations and report scripts. If substitution variables with the same name exist at server, application, and database levels, the order of precedence for the variables is as follows: a database level substitution variable supersedes an application level variable, which supersedes a server level variable.
drop variable	Remove a substitution variable and its corresponding value from the application.
rename to	Rename the application. When you rename an application, the application and the application directory (<code>ARBORPATH\App\application_name</code>) are renamed.
sync user	Synchronize the named user's information on this Essbase application with the latest matching user information found on Shared Services. To issue this statement, you must be an Administrator, Application Manager, or Database Manager.
sync group	Synchronize the named group's information on this Essbase application with the latest matching group information found on Shared Services. To issue this statement, you must be an Administrator, Application Manager, or Database Manager.
sync all_users_groups	Synchronize all user and group information on this Essbase application with the latest user and group information found on Shared Services. To issue this statement, you must be an Administrator, Application Manager, or Database Manager.
reregister	Re-establish this Essbase application as a Shared Services application. This statement reregisters the application with Shared Services, in the event that you have: <ul style="list-style-type: none"> ● deleted the application from Shared Services but kept using it in Essbase. ● changed the Essbase Administration Server location, name, or port number. ● changed the Essbase Server name or port number. <p>To issue this statement, you must be an Administrator or Application Manager.</p>

Keyword	Description
all reregister	<p>Re-establish this and all other Essbase applications as Shared Services applications. This statement reregisters the applications with Shared Services, in the event that you have:</p> <ul style="list-style-type: none"> • deleted the application from Shared Services but kept using it in Essbase. • changed the Essbase Administration Server location, name, or port number. • changed the Essbase Server name or port number. <p>To issue this statement, you must be an Administrator or Application Manager on all applications; for any applications for which you do not have sufficient permissions, the re-registration will be skipped with a warning.</p>

Example

```
alter application Sample set minimum permission read;
```

Grants all users read access to all databases in the Sample application. Users can retrieve data values and run report scripts.

```
alter application Sample disable commands;
```

Prevents all users from making requests to the application scope. Use this statement before performing application-wide update and maintenance operations.

```
alter application Acme set variable Current_month July;
```

Assigns the string value July to the substitution variable "Current_month."
"Current_month" may be referenced by calculations in the Acme application.

Alter Database

[Click here for aggregate storage version](#)

Select a subset of **alter database**:

- [Alter Database enable | disable](#)
- [Alter Database Set](#)
- [Alter Database \(Misc\)](#)
- [Alter Database \(disk volumes\)](#)

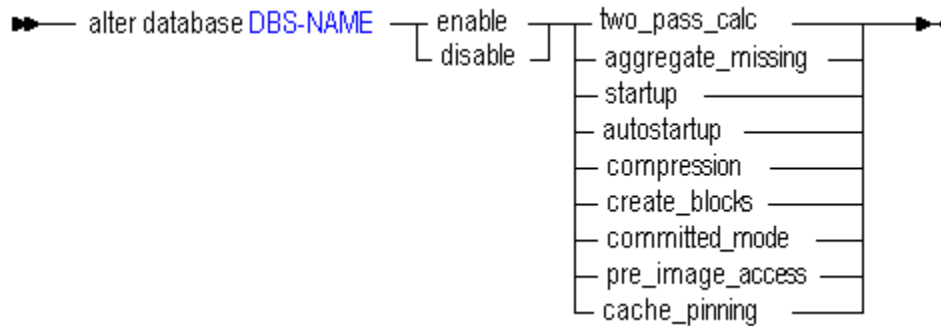
Alter Database enable | disable

[Click here for aggregate storage version](#)

Change database-wide settings.

Permission required: create_application.

Syntax



Use `alter database` to change the following database-wide settings:

Keyword	Description
enable two_pass_calc	<p>Recalculate (after a default calculation) database outline members tagged as Two Pass, so they will be recalculated after other database members have been consolidated. This setting is enabled by default.</p> <p>Members that usually require a two-pass calculation are those members of the Accounts dimension that are calculated by a formula rather than by hierarchical consolidation. These members are typically ratios, such as "Profit % Sales" (profit percentage of sales), which has a member formula.</p> <p>This setting is ignored during a calculation script; it is used only during a default calculation. To use two-pass calculation in a non-default calculation, use the <code>CALC TWOPASS</code> command in the calculation script.</p>
disable two_pass_calc	<p>Do not recalculate database outline members tagged as Two Pass after a default calculation. Two-pass calculation is enabled by default.</p>
enable aggregate_missing	<p>Consolidate #MISSING values along with the regular database consolidation. If you never load data at parent levels, aggregating #MISSING values can improve calculation performance, depending on the ratio between upper level blocks and input blocks in the database.</p> <p>If this setting is enabled and you load values directly at the parent level, these parent-level values will be replaced by the results of the consolidation, even if the results are #MISSING values. The aggregate missing setting is disabled by default.</p>
disable aggregate_missing	<p>Do not consolidate #MISSING values. This is the default. Data that is loaded at parent levels is not overwritten by #MISSING values of children below it. However, if any of the child data values are not #MISSING, these values are consolidated and overwrite the parent values.</p>
enable startup	<p>Enable users to start the database directly or as a result of requests requiring the database to be started. Startup is enabled by default.</p>
disable startup	<p>Prevent all users from starting the database directly or as a result of requests that would start the database. Startup is enabled by default.</p>
enable autostartup	<p>Automatically start the database when the application to which it belongs starts. Autostartup is enabled by default. This setting is applicable only when startup is enabled.</p>
disable autostartup	<p>Prevent automatic starting of the database when the application to which it belongs starts. Autostartup is enabled by default.</p>
enable compression	<p>Enable data compression. By default, Bitmap compression is enabled. To switch to a different compression type, use <code>alter database set compression</code>.</p>

Keyword	Description
disable compression	Disable data compression. By default, Bitmap compression is enabled.
enable create_blocks	<p>Allow Essbase to create a data block when you assign a non-constant value to a member combination for which a data block does not already exist. Block creation on equation is disabled by default, because it can result in a very large database.</p> <p>When you assign a constant to a member on a sparse dimension, you do not need to enable Create Blocks on Equation, because Essbase would create a data block anyway. For example, "West = 5 ; " would result in the creation of data blocks, with or without the Create Blocks on Equation setting enabled.</p> <p>You do need to check this option if you want blocks created when you assign anything other than a constant to a member on a sparse dimension for which a data block does not already exist. For example, if no data exists for Actuals, a member of a sparse Scenario dimension, then you need to enable Create Blocks on Equation in order to perform the following allocation:</p> <pre>2002Forecast = Actuals * 1.05;.</pre>
disable create_blocks	Turn off the Create Blocks on Equation setting. The setting is disabled by default.
enable committed_mode	Set the database isolation level to committed access, meaning that only one transaction at a time can update data blocks. Essbase holds read/write locks on all data blocks until the transaction and the commit operations are performed. If pre-image access is enabled, users (or transactions) can still have read-only access to data at its last commit point. For more information, see the <code>enable pre_image_access</code> setting. The default isolation-level mode is Uncommitted.
disable committed_mode	Turn off the Committed Mode setting, reverting to the default isolation level of Uncommitted for the database.
	<p>Note: Spreadsheet Add-in lock and send operations are always in committed mode.</p> <p>In uncommitted mode, Essbase allows transactions to hold read/write locks on a block-by-block basis. Essbase releases a block after it is updated, but does not commit blocks until the transaction is completed, or until a specified number of blocks or rows (a "synchronization point") has been reached. You can set this limit using the <code>implicit_commit</code> settings.</p>
enable pre_image_access	<p>Allow users (or other transactions) read-only access to data at its last commit point, when the database is in committed mode (meaning that data blocks may be locked for the duration of a concurrent transaction). Pre-image access is enabled by default when the database is in committed mode.</p> <p>See also the <code>enable committed_mode</code> setting.</p>
disable pre_image_access	Disable pre-image access, disallowing read-only access to locked blocks of data at their last commit point (this setting is only applicable while the database is in committed mode). Pre-image access is enabled by default when the database is in committed mode.

Keyword	Description
enable cache_pinning	<p>Enable cache memory locking, which locks the memory used for the index cache, data file cache, and data cache into physical memory, giving the Essbase Server kernel priority use of system RAM. Cache memory locking improves performance for a database because the system memory manager does not need to swap the memory used by the caches when swapping the memory used by the Essbase Server. The setting takes effect after you restart the database.</p> <p>By default, cache memory locking is disabled. To use cache memory locking, you must be using direct I/O (buffered I/O is the default). For more information, see the <i>Oracle Essbase Technical Reference</i> documentation for the DIRECTIO setting for <code>essbase.cfg</code>.</p>
disable cache_pinning	<p>Disable cache memory locking, reverting to the default.</p>

Example

```
alter database Sample.Basic enable cache_pinning;
```

Locks database cache pages in physical memory so that the operating system will not page them out while the database is still using them.

```
alter database Sample.Basic disable two_pass_calc;
```

Prevents recalculation (after a default calculation) of members tagged as Two Pass.

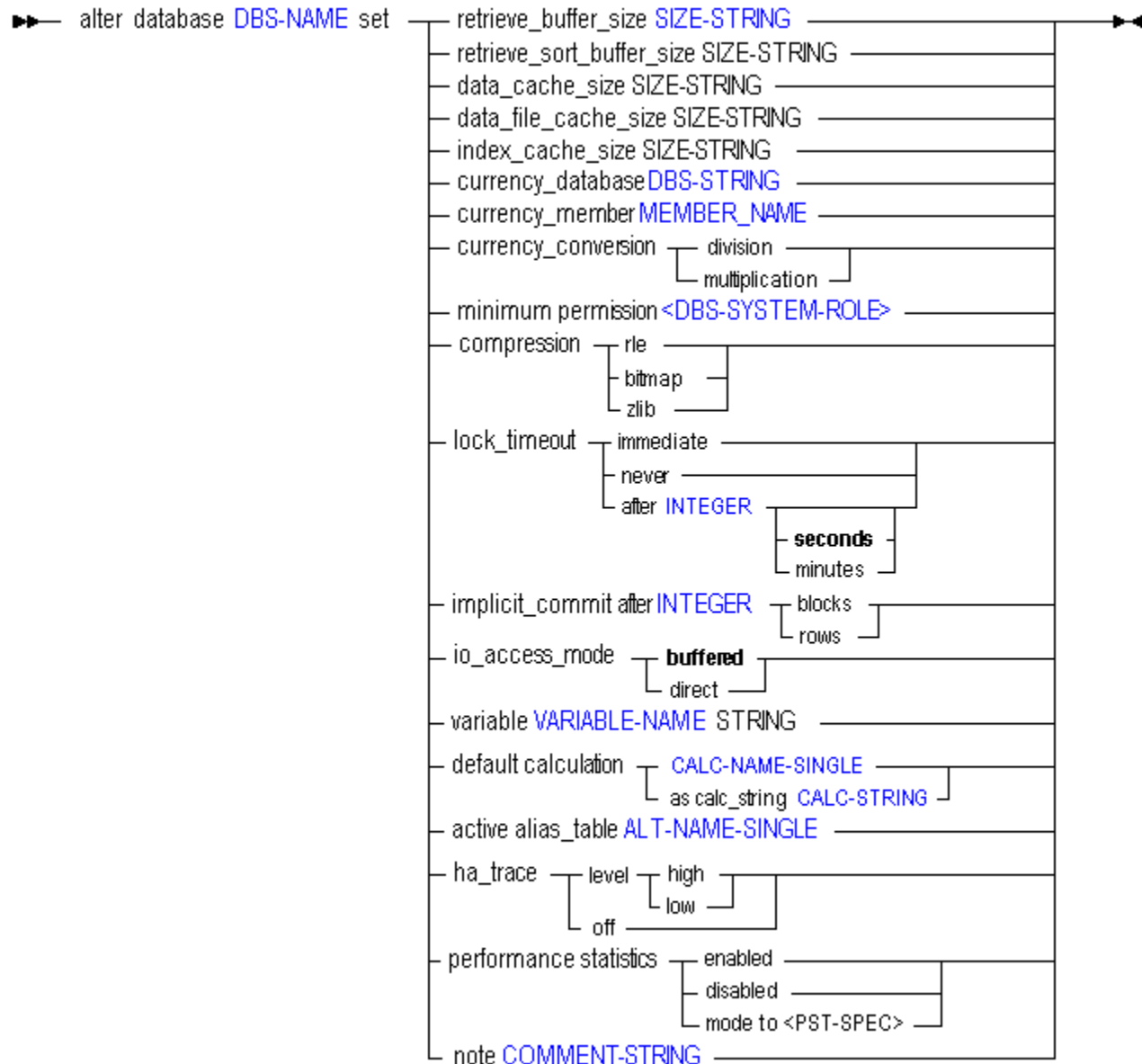
Alter Database Set

[Click here for aggregate storage version](#)

Change database-wide settings.

Permission required: `create_application`.

Syntax



Use `alter database set` to change the following database-wide settings:

Keyword	Description
<code>retrieve_buffer_size</code>	Change the database retrieval buffer size. This buffer holds extracted row data cells before they are evaluated by the <code>RESTRICT</code> or <code>TOP/BOTTOM</code> Report Writer commands. The default size is 10 KB (on 32-bit platforms), and 20 KB (on 64-bit platforms). The minimum size is 2 KB. Increasing the size may improve retrieval performance.
<code>retrieve_sort_buffer_size</code>	Change the database retrieval sort buffer size. This buffer holds data until it is sorted. The Report Writer and Essbase Query Designer use the retrieval sort buffer. The default size is 10 KB (on 32-bit platforms), and 20 KB (on 64-bit platforms). The minimum size is 2 KB. Increasing the size may improve retrieval performance.

Keyword	Description
data_cache_size	Change the data cache size. The data cache is a buffer in memory that holds uncompressed data blocks. Essbase Server allocates memory to the data cache during data load, calculation, and retrieval operations as needed. The default and minimum size is 3072 KB.
data_file_cache_size	Change the data file cache size. The data file cache is a buffer in memory that holds compressed data files (.PAG files). Essbase Server allocates memory to the data file cache during data load, calculation, and retrieval operations as needed. The data file cache is not used when buffered I/O is used; you must use direct i/o to use the data file cache. The default size is 32 MB.
index_cache_size	Change the index cache size. The index cache is a buffer in memory that holds index pages. When a data block is requested, Essbase looks at the index pages in the index cache to find its location on disk. The default size is 1 MB when buffered I/O is used, and 10 MB when direct I/O is used. Buffered I/O is the default for this release.
currency_database	Link the database with a currency database. A currency database enables you to convert currency values in a database from one currency into another currency.
currency_member	Specify the member to use as a default value in currency conversions. You can specify any valid member of the dimension defined as "Currency Type" in the currency database.
currency_conversion	Specify whether during currency conversion, the calculation method multiplies the currency database exchange rates with the main database values, or that the currency database exchange rates are divided by the main database values.
minimum permission	Set a level of permission that all users or groups can have to the database. Users or groups with higher granted permissions than the minimum permission are not affected.
compression rle	Set the database to use run-length encoding (RLE) compression. Essbase compresses repetitive, consecutive values, including zeros and #MISSING values. The default compression type is bitmap. When a compressed data block is brought into the data cache, Essbase expands the block to its full size, regardless of the scheme that was used to compress it.
compression bitmap	Set the database to use bitmap compression, the default. Essbase stores only non-missing values and uses a bitmapping scheme. When a compressed data block is brought into the data cache, Essbase expands the block to its full size, regardless of the scheme that was used to compress it.
compression zlib	Set the database to use ZLIB compression. When a compressed data block is brought into the data cache, Essbase expands the block to its full size, regardless of the scheme that was used to compress it. If your database allows or requires "Aggregate Missing Values" setting set to YES, then you may want to consider using ZLIB as the compression scheme. ZLIB particularly works well on such databases compared to other compression schemes. However, changing the aggregate missing values setting may have an impact on calculation results - see the <i>Oracle Essbase Database Administrator's Guide</i> . Consider using ZLIB only if you have already determined that the setting should be YES for other reasons.
lock_timeout	Change the interval to wait for blocks to be unlocked when the database is in committed mode. If a transaction request is made that cannot be granted in the allotted time, the transaction is rolled back until a lock can be granted.

Note: Spreadsheet Add-in lock and send operations are always in committed mode.

Keyword	Description
implicit_commit after <number> blocks	When uncommitted access is enabled, set the frequency at which Essbase commits data blocks (after the specified number of blocks has been reached).
implicit_commit after <number> rows	When uncommitted access is enabled, set the frequency at which Essbase commits data blocks (after the specified number of rows has been reached).
io_access_mode	<p>Change the input/output setting you wish to use for the database. The change takes effect the next time the database is started.</p> <p>Buffered I/O uses the file system's buffer cache, and is the default.</p> <p>Direct I/O bypasses the file system's buffer cache, and is able to perform asynchronous, overlapped I/Os, providing faster response time and more potential to optimize cache sizes for databases.</p> <p>If you set a database to use direct I/O, Essbase will attempt to use direct I/O each time the database is started. If direct I/O is not available on your platform at the time the database is started, Essbase will use buffered I/O, which is the default.</p>
variable	Change the value of an existing substitution variable on the database. The value must not exceed 256 bytes. It may contain any character except a leading ampersand (&).
default calculation	Change the default calculation (which, by default, is <code>CALC ALL;</code>) to the stored calculation script you specify, or to an anonymous (unstored) calculation string.
active alias_table	Set an alias table as the primary table for reporting and any additional alias requests. Only one alias table can be used at a time. This setting is user-specific; it only sets the active alias table for the user issuing the statement.
ha_trace level	<p>Enable logging of queries generated by Hybrid Analysis operations, such as running a report involving relationally stored members, or drilling into a spreadsheet containing relationally stored members. The queries are logged into the file <code>essha.log</code>, which is found in the root Essbase installation directory, <code>Essbase</code>.</p> <p>The level option controls the amount of information written to <code>essha.log</code>. Level <code>high</code> should be used only for debugging purposes and should be rarely used because it can quickly fill up the log file. Level <code>low</code> is recommended.</p>
ha_trace off	Turn off logging of queries generated by Hybrid Analysis operations.
performance statistics enabled	Turn on performance-statistics gathering. You might do this when you want to tune the system, change hardware configuration, or monitor I/O. The measurement begins for current processes as soon as you enable it. Any subsequent queries for statistics return measurements spanning from the time of enablement to the time of the query. Performance statistics can be retrieved using query database .
performance statistics disabled	Turn off performance-statistics gathering. This halts the collection of statistics; it does not prevent anyone from retrieving old statistics using query database .

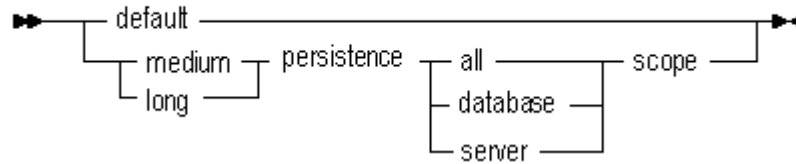
Keyword

performance statistics mode to <PST-SPEC>

Description

Reset [performance statistics](#) gathering for a specified persistence and scope. Each of the statistics tables available using `query database` has a pre-defined persistence and scope. When you use `set performance statistics mode`, you select the persistence and scope to reset, and the collecting of measurements starts over for the applicable tables.

<PST-SPEC> ::=



note

Create an informational note about the database that Spreadsheet Add-in users can see from the login dialog box. For example, 'Calc in progress: do not update.'
Database notes can be up to 64 kilobytes long.

Example

```
alter database Sample.Basic set lock_timeout after 120;
```

Changes the number of seconds to wait for blocks to be unlocked. If a transaction request is made which cannot be granted in 120 seconds, the transaction is rolled back until a lock can be granted.

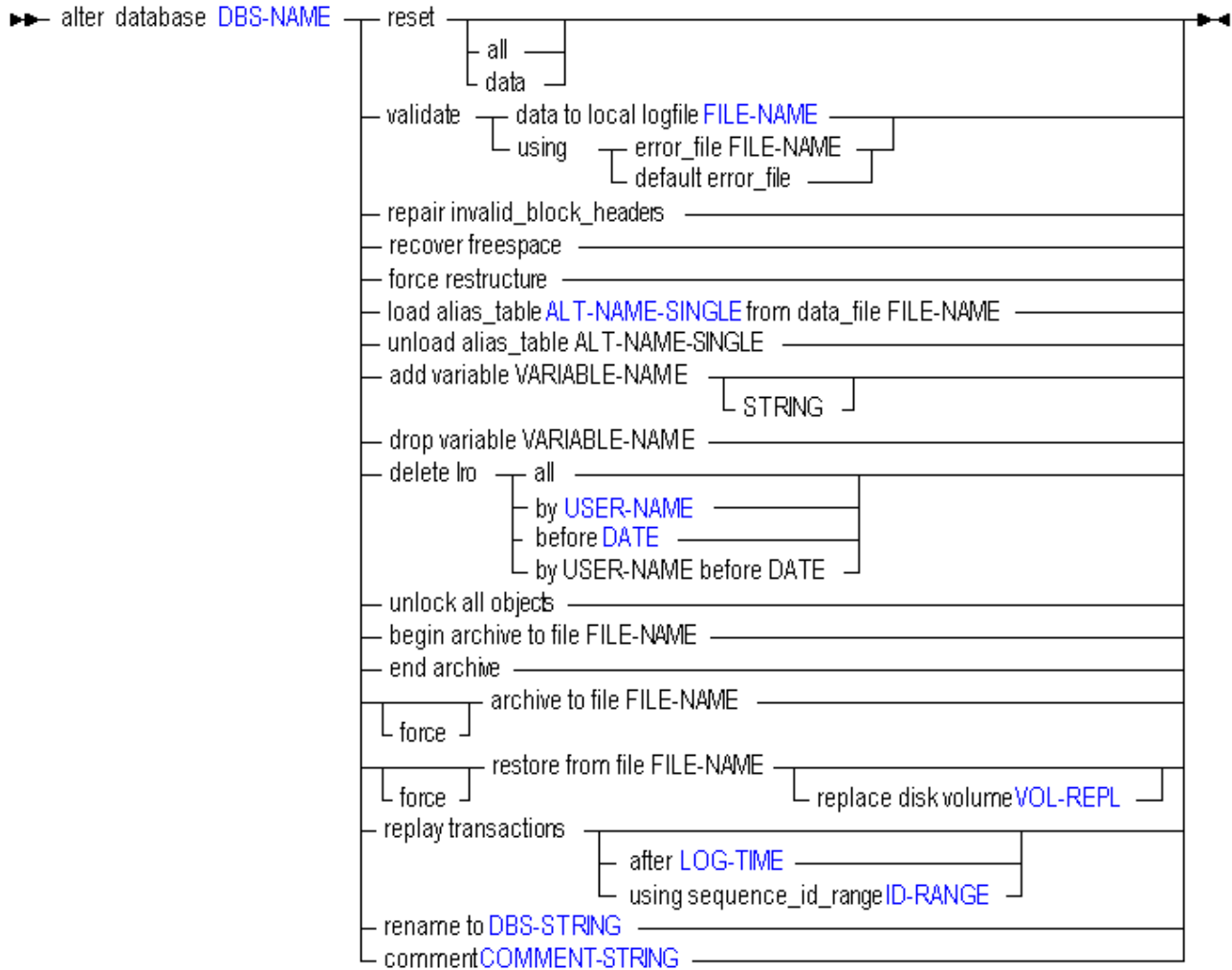
Alter Database (Misc)

[Click here for aggregate storage version](#)

Change database-wide settings.

Permission required: `create_application`.

Syntax



Use `alter database` to change the following database-wide settings:

Keyword	Description
reset	Clear all data and linked-reporting objects from the database, but preserve the outline.
reset all	Clear all data, Linked Reporting Objects, and the outline.
reset data	Same as using <code>reset</code> .

Keyword	Description
validate data to local logfile...	<p>Create a local log file with all index combinations for which blocks contain invalid block headers.</p> <p>Before using this MaxL statement, be sure that the server is not performing other operations, such as calculations or data loads; otherwise, an exception error may occur.</p> <p>The recommended procedure is:</p> <ol style="list-style-type: none"> 1. Disable all logins. 2. Forcibly log off all users. 3. Run the MaxL statement to get invalid block header information. 4. Repair invalid block headers, if applicable. <p>For example,</p> <pre>alter application sample disable connects; alter system logout session on database sample.basic; alter database sample.basic validate data to local logfile 'invalid_blocks'; alter database sample.basic repair invalid_block_headers;</pre>
validate using...	<p>Check the database for data and structural integrity. A file is created containing error messages if there are problems. The default error file is <code>VALIDATE.LST</code> in the application or database directory. For example:</p> <pre>Hyperion/products/Essbase/EssbaseServer/app/sample/basic/ VALIDATE.LST.</pre> <p>The validate utility verifies the following:</p> <ul style="list-style-type: none"> ● That blocks, sections, block type, block length, and floating-point numbers are valid. ● That the index contains an entry for every data block. ● That keys in the index page are matched with keys in the corresponding data blocks. Keys out of order indicate corruption. ● Structural integrity of index freespace information. ● Structural integrity of the LRO catalog.
repair invalid_block_headers	<p>Delete all blocks that have invalid headers. Before using this statement, see <i>validate data to local logfile</i>.</p>
recover freespace	<p>Explicitly recover database freespace in the event of a crash or abnormal shutdown. Beginning with Release 7.0, freespace recovery only occurs if you explicitly request it.</p>
force restructure	<p>Explicitly restructure the database to eliminate or reduce fragmentation.</p>

Keyword	Description
load alias_table	<p>Load an alias table from a file to the current database. The feeder file (FILE-NAME) must follow these rules:</p> <ul style="list-style-type: none"> ● Must be correctly formatted. ● Must be located on the Essbase Server computer, not on a client computer. ● FILE-NAME must include the full path. <p>Sample contents of a feeder file for loading an alias table:</p> <pre style="margin-left: 40px;">\$ALT_NAME "400-10" Guava "400-20" Tangerine "400-30" Mango \$END</pre>
unload alias_table	Delete the specified alias table.
add variable	<p>Create a database-level substitution variable by name, and optionally assign a string value for the variable to represent. You can assign or change the value later using <code>set variable</code>. A substitution variable acts as a global placeholder for information that changes regularly. Substitution variables may be referenced by calculations and report scripts.</p> <p>If substitution variables with the same name exist at server, application, and database levels, the order of precedence for the variables is as follows: a database level substitution variable supersedes an application level variable, which supersedes a server level variable.</p>
drop variable	Remove a substitution variable and its corresponding value from the database.
delete lro	Delete Linked Reporting Objects linked to the active database for a given user name or modification date.
unlock all objects	Unlock all objects on the database that are in use by a user or process.
begin archive to file	<p>Prepare the database for backup by an archiving program, and prevent writing to the files during backup. This statement requires the database to be started.</p> <p>Begin archive achieves the following outcomes:</p> <ul style="list-style-type: none"> ● Commits any modified data to disk. ● Switches the database to read-only mode. The read-only state persists, even after the application is restarted, until it is changed back to read-write using <code>end archive</code>. ● Reopens the database files in shared, read-only mode. ● Creates a file containing a list of files that need to be backed up. Unless a different path is specified, the file is stored in the database directory. <p>Begin archive and end archive do not perform the backup; they simply protect the database during the backup process.</p>
end archive	<p>Return the database to read-write mode after backing up the database files.</p> <p>This statement requires the database to be started.</p> <p>End archive achieves the following outcomes:</p> <ul style="list-style-type: none"> ● Returns the database to read-write mode. ● Re-opens database files in exclusive, read-write mode.

Keyword	Description
archive to file	<p>Write a copy of the database files to a specified archive file that resides on the Essbase Server computer. Provide the full pathname to an existing directory and the name of the archive file. If only the archive filename is provided, Essbase writes the archive file to <i>ARBORPATH/app</i>.</p> <p>Oracle recommends writing the archive file to a different disk than the one where <i>ARBORPATH</i> is located and recommends that you name the file with a <i>.arc</i> extension.</p> <p>By default, Essbase creates a single, large archive file. The size of the archive file corresponds to the size of the database you back up and is limited only by disk space. If, however, in your environment you cannot use large files or the file-transfer tools that you use cannot handle large files, you can configure Essbase to split the archive file into multiple files, with each file no larger than 2 GB. In the <i>essbase.cfg</i> file, set the “SPLITARCHIVEFILE” on page 497 configuration setting to TRUE.</p> <p>Note: If you use the single-file configuration, Oracle recommends saving archive files to a file system that supports large files. For Windows, the file system must be formatted as NTFS. For UNIX, large file support must be enabled (for example, use the <i>ULIMIT</i> setting to specify a specific file size based on the size of the database or set <i>ULIMIT</i> to unlimited). See your operating system documentation.</p> <p>If you are backing up a database to an existing archive file, you must use the force archive to file grammar to overwrite the file.</p> <hr/> <p>Caution! When using the force option, be sure that you no longer need the contents of the existing archive file.</p>
force archive to file	<p>Overwrite the contents of an existing archive file.</p> <hr/> <p>Caution! When using the force option, be sure that you no longer need the contents of the existing archive file.</p>
restore from file	<p>Restore a database with the contents of the specified archive file.</p> <p>If you have configured Essbase to split the archive file into multiple files (“SPLITARCHIVEFILE” on page 497), you only need to specify the filename of the main archive file that you want to restore (for example, <i>samplebasic.arc</i>). All archive files must reside in the same directory as the main archive file.</p> <p>Typically, you restore a database to the application and database from which the backup was taken and, therefore, the names of the backed up and restored database and its associated application are the same. If, however, the names of the backed up database and application are not the same as the application and database to which you are restoring data, you must use the force restore from file grammar.</p>

Keyword	Description
restore from file...replace disk volume VOL-REPL	<p>Restore a database with the contents of the specified archive file and replace the specified disk volumes.</p> <p>Valid values for the VOL-REPL argument are a comma-separated list of volumes to replace:</p> <ul style="list-style-type: none"> ● 'VOL1' with 'VOL2' ● 'VOL3' with 'VOL4' ● 'VOL5' with 'VOL6' <p>The number of disk volumes used and the space required for the restored database must be the same as for the database before it was backed up. Only the name of disk volumes can be different.</p>
force restore from file...	<p>Use the contents of the specified archive file to restore to a database that has different names than the archived database or its associated application. For example, you can use the archive file for Sample.Basic to restore to Sample.New (the database name is different), MyCompany.Basic (the application name is different), or MyCompany.New (both names are different).</p>
replay transactions	<p>Replays the database transactions that were logged after the last replay request was originally executed or after the last restored backup's time (whichever occurred later).</p> <p>Transactions that are executed and logged after the restore operation are not replayed, unless you replay those transactions using their sequence IDs. After restoring a database, Oracle recommends that you finish replaying the transactions that were logged after the backup and before the restore and that are needed to fully recover the database; then you can continue executing new transactions.</p>
replay transactions after LOG-TIME	<p>Replays the transactions that were logged after the specified time. Enclose the TIME value in quotation marks; for example: '11_20_2007:12:20:00'</p>
replay transactions using sequence_id_range ID-RANGE	<p>Replays the transactions specified by a comma-separated list of sequence ID ranges. A range can consist of:</p> <ul style="list-style-type: none"> ● A single transaction: <i>n</i> to <i>n</i>; for example, 1 to 1 ● Multiple transactions: <i>x</i> to <i>y</i>; for example, 20 to 100 <p>Each logged transaction is assigned a sequence ID, indicating the order in which the transaction was performed. To ensure the integrity of the restored data after a replay, Essbase enforces the replay of transactions in the same order in which they were originally performed. The order of sequence IDs are tracked across multiple replay commands.</p> <p>Note: You can skip replaying a transaction if you are absolutely sure that the transaction results are not required to recover the database.</p>
rename to	<p>Rename the database. When you rename a database, the database directory is also renamed.</p>
comment	<p>Create a description of the database. The maximum number of characters is 80. This description is available to database administrators. To annotate the database for Spreadsheet Add-in users, use <code>set note</code>.</p>

Example

```
alter database Sample.Basic archive to file /Hyperion/samplebasic.arc;
```

Backs up Sample.Basic database files to the specified archive file (`samplebasic.arc`) on Essbase Server.

```
alter database Sample.Basic force archive to file /Hyperion/samplebasic.arc;
```

In backing up the Sample.Basic database files, overwrites the existing archive file (`samplebasic.arc`).

```
alter database Sample.Basic restore from file /Hyperion/samplebasic.arc;
```

Restores the Sample.Basic database using the `samplebasic.arc` archive file.

```
alter database MyCompany.New force restore from file /Hyperion/samplebasic.arc;
```

Uses the archive file for the Sample.Basic database (`samplebasic.arc`) to restore the MyCompany.New database.

```
alter database Sample.Basic restore from file /Hyperion/samplebasic.arc replace disk volume 'C' with 'F', 'D' with 'G', 'E' with 'H';
```

Restores the Sample.Basic database using the `samplebasic.arc` archive file and replaces the specified disk volumes.

```
alter database Sample.Basic replay transactions using sequence_id_range 1 to 10,20 to 100;
```

Replays the transactions in the Sample.Basic database with sequence IDs 1 through 10 and 20 through 100.

```
alter database Sample.Basic replay transactions after '11_20_2007:12:20:00';
```

Replays all transactions that were logged after the specified time.

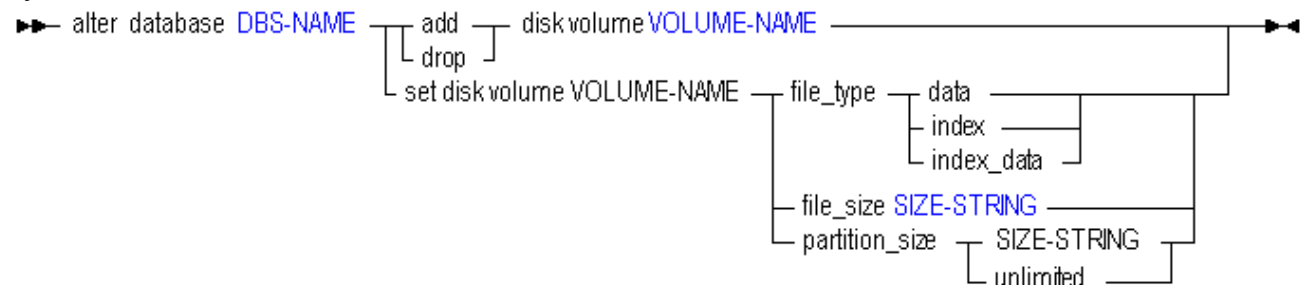
See Also

- [Alter Database enable | disable](#)
- [Alter Database Set](#)
- [Alter Database \(disk volumes\)](#)

Alter Database (disk volumes)

Add, delete, or modify a database disk volume. Permission required: `create_application`. Disk volumes apply only to block storage databases.

Syntax



Use `alter database` to change the following database disk volume settings:

Keyword	Description
add disk volume	Add a disk volume definition if you want to allocate storage across multiple volumes, or restrict space used on a volume. After adding a disk volume definition, use <code>set disk volume</code> to place restrictions on files stored on the disk volume.
drop disk volume	Remove a disk volume definition. If no disk volume is defined, data and index files are stored in the database directory (for example, <code>\$ARBORPATH/app/sample/basic</code>).
set disk volume	Specify what types of files should be stored on the disk volume. You can allocate storage for index files, data files, or both. You can specify the maximum file size and partition size allowed on the disk volume.

Notes

Add a disk volume definition if you want to allocate storage across multiple volumes, or restrict space used on a volume. You can allocate storage for index files, data files, or both.

Files are written to the disk volume in the following directory structure:

```
.../app/app_name/db_name
```

For new files, disk volume settings become effective after the database is restarted. Previously existing files and volumes are not affected.

If no disk volume is defined, data and index files are stored in the database directory (for example, `$ARBORPATH/app/sample/basic`).

`File_size` is the maximum size an index or data file may attain. Default = 2G; minimum = 8192K (8M).

`Partition_size` is the maximum amount of disk space allocated to the volume. Default = unlimited.

Example

```
alter database Sample.Basic set disk volume c file_type index;
```

Changes the storage settings for Sample Basic so that the alternate disk volume specified as the C: drive stores only index files.

Alter Drillthrough

Edit drill-through URL definitions used to link to content hosted on Oracle ERP and EPM applications.

Syntax

```
▶▶ alter drillthrough URL-NAME from xml_file FILE-NAME ▶▶
▶ on { MEMBER-EXPRESSION } [ allow_merge ]
```

Use `alter drillthrough` to edit a URL definition in the following ways:

Keyword	Description
alter drillthrough	Edit drill-through URL metadata. The number of drill-through URLs per database is limited to 255.
from xml_file	Indicate the path to the local URL XML file that defines the link information. The URL XML is created by the ERP or EPM application that deployed the Essbase database. The XML contains the drill-through URL display name as well as a URL enabling the hyperlink from a cell to a Web interface to occur. For a sample URL XML file, see Create Drillthrough .
on {<member-expression>,...}	Define the list of drillable regions, using the same Essbase member-set calculation language that is used to define security filters. The list of drillable regions must be enclosed in {brackets}. The number of drillable regions in a drill-through URL is limited to 256. The number of characters per drillable region is limited to 65536.
allow_merge	Optional: Merge the drillable-region definition instead of replacing it on update.

Example

```
alter drillthrough sample.basic.myURL from xml_file "C:/drillthrough/data/myfile.xml" on
{'@Ichildren("Qtr1")', '@Ichildren("Qtr2")'} allow_merge;
```

See Also

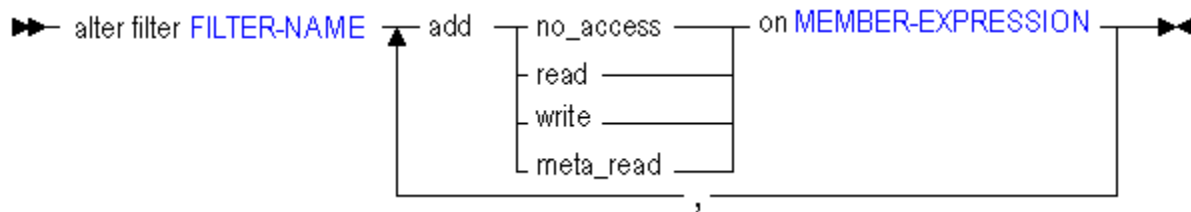
- [create drillthrough](#)
- [display drillthrough](#)
- [drop drillthrough](#)

Alter Filter

Add filter rows to a database security filter. Filters control security for database objects. Use [grant](#) to assign filters to users and groups.

Minimum permission required: Database Manager.

Syntax



Use **alter filter** in the following ways to edit a filter:

Keyword	Description
alter filter ...add no_access on <member-expression>	Block access to a specified member combination.

Keyword	Description
alter filter ... add read on <member-expression>	Provide read-only access to a specified member combination.
alter filter ... add write on <member-expression>	Provide write access to a specified member combination.
alter filter ... add meta_read on <member-expression>	Restrict access to siblings and ancestors of the member expression. In case of a filtering conflict, the MetaRead filtering overrides the other filter permissions. For more information about metadata filtering, see “Metadata Filtering” on page 926 .

Notes

- Filters created using MaxL must be valid. For information about filter syntax, see the *Oracle Essbase Database Administrator's Guide*.
- MEMBER-EXPRESSION must be enclosed in single quotation marks. It can be a comma-separated list.

Example

```
alter filter sample.basic.filt7 add read on '@Descendants("East")';
```

Adds a row to a Sample Basic filter named filt7, giving read-only access to the data for the eastern states.

```
alter filter sample.basic.filt8 add read on '@Descendants("East")', add write on '@Descendants("West")';
```

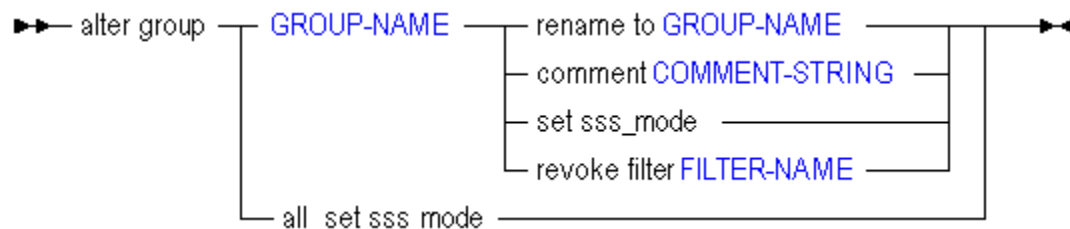
Adds two rows to a Sample Basic filter named filt8.

Alter Group

Rename a group or change the comment that describes the group.

Permission required: create_user.

Syntax



Use **alter group** to change the following settings. See also **alter user**.

Keyword	Description
rename to	Rename the group.

Keyword	Description
comment	Create a description of the group.
set sss_mode	Migrate the group to Oracle Hyperion Enterprise Performance Management System security mode. This might be useful if the group migration failed using alter system . Minimum permission required: Administrator. For more information, see the <i>Oracle Essbase Database Administrator's Guide</i> chapter titled "User Management and Security."
revoke filter	Remove a filter assignment to this group. Privilege required: Application manger.
	Note: This statement does not remove filter assignments granted to individual users. To remove filter assignments to users, use Alter User .
all set sss_mode	Same as set sss_mode, but for all groups.

Notes

See Notes for [Alter User](#).

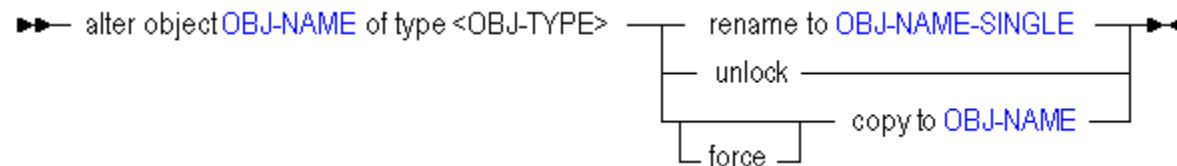
Example

```
alter group NewGroup rename to Recruit;
alter group Recruit comment 'This group is for the newly hired';
alter group MyGroup set sss_mode;
```

Alter Object

Rename, unlock, or copy a database-related artifact.

Syntax



Use **alter object** to edit artifacts in the following ways:

Keyword	Description
rename to	Rename the artifact. Not applicable for partition files, worksheets, or outlines.
unlock	Unlock an artifact that is locked by another user or process. Not applicable for alias tables and worksheets. Unlocking an artifact of type LRO is applicable for stored linked-reporting objects only; that is, files with the .LRO extension.
	Note: To unlock all database artifacts, use <code>alter database DBS-NAME unlock all objects;</code>
copy to	Make a copy of a server artifact. Not applicable for partition files, worksheets, or outlines. If an artifact of the new name already exists, it is replaced.

Keyword **Description**

force copy Make a copy of a server artifact. Not applicable for partition files, worksheets, or outlines. If an artifact of to the new name already exists, it is replaced. If an administrator issues the statement with the force keyword, locked artifacts are unlocked, copied, and re-locked.

Notes

- Specified artifacts must be persisted in the database directory.
- To copy artifacts that are not persisted in the database directory, use the [EXPORT ESSCMD](#) command.
- Attempting to rename or copy an artifact of type "partition_file" returns an error.

Example

```
alter object sample.basic.genref of type rules_file rename to 'level';
```

Renames a rules file in the Sample Basic directory, named `genref.rul`, to `level.rul`.

```
alter object sample.basic.Calcdat of type text rename to 'c_data';
```

Renames a text file in the Sample Basic directory, named `calcdat.txt`, to `c_data.txt`.

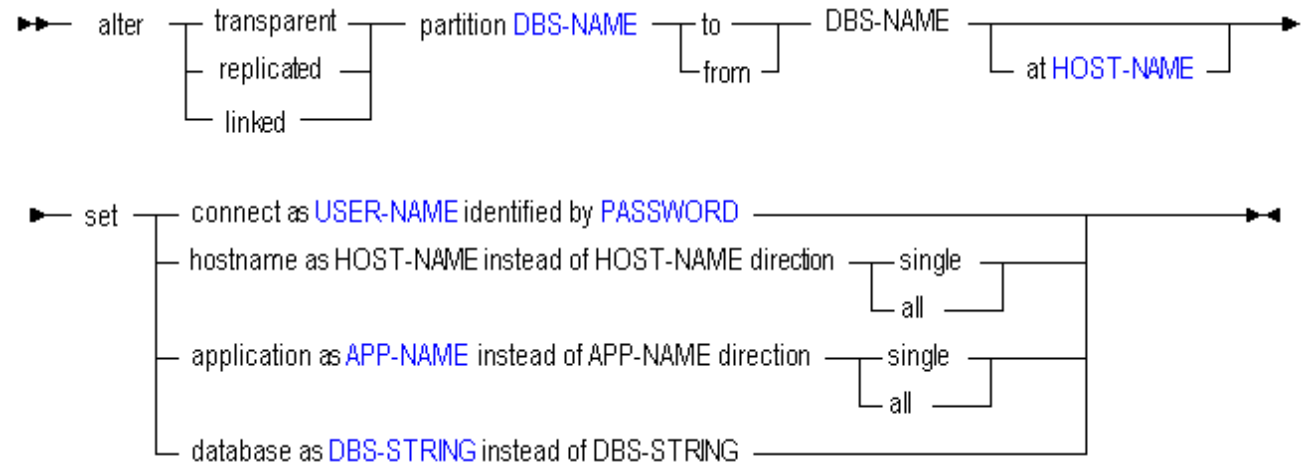
```
alter object samppart.company.company of type partition_file unlock;
```

Unlocks the partition definition file for the Samppart Company database.

Alter Partition

Fix invalid or dangling partition references. Change the authorized user who can connect to both databases. Change the name of an application, database, or host (in the event that something was renamed).

Syntax



Use `alter partition` to edit partitions in the following ways:

Keyword	Description
...set connect	Change the user authorized to access the partitioned databases.
...set hostname	Edit the partition definition to include the correct computer name that hosts the partition source database, target database, or both.
...set application as	Edit the partition definition to include a corrected application name. This is useful if <i>one</i> application name was changed; if both application names changed, the partition definition cannot be corrected and you must re-create it.
...set database as	Edit the partition definition to include a corrected database name. This is useful if <i>one</i> database name was changed; if both database names changed, the partition definition cannot be corrected and you must re-create it.
...direction single	See Example 2 (Alter Partition) , Example 4 (Alter Partition) , , and Example 5 (Alter Partition) .
...direction all	See Example 3 (Alter Partition) .

Notes

- The first DBS-NAME is the local database, and the second DBS-NAME is the remote database.
- Directing a partition *to* the remote site means the current database is the source. Creating a partition *from* the remote site means the current database is the target.
- To change the authorized partition user, you must change the user for both partitioned databases, as shown in [Example 1 \(Alter Partition\)](#).
- If a partitioned host, application, or database is renamed, the rename does not propagate to the partition definition, so you must use alter partition to change the name in the partition definition. As shown in Examples 2 through 5, you must give the old name and the new name. If both names were changed, the partition definition is not recoverable, and must be re-created.

Example

Example 1 (Alter Partition)

The following example changes the user authorized to access the partitioned databases.

```
/* To change authorized partition user on target, log in to source & then use: */
alter transparent partition app1.source to app2.target
set connect as newuser identified by newpasswd;

/* To change authorized partition user on source, log in to target & then use: */
alter transparent partition app2.target from app1.source
set connect as newuser identified by newpasswd;
```

Example 2 (Alter Partition)

In the following example, alter partition is used to fix a partition definition that became invalid when a host name (oldHost) changed and affected only one half of the partition definition (app2.target):

```
alter transparent partition app1.source to app2.target at oldHOST
    set hostname as newHOST instead of oldHOST direction single;
```

where `direction single` indicates that only the target host name needs to be changed.

Example 3 (Alter Partition)

In the following example, `alter partition` is used to fix a partition definition that became invalid when a host-name change affected both the source and the target, because both applications were on the same host:

```
alter transparent partition app1.source to app1.target at newHOST
    set hostname as newHOST instead of oldHOST direction all;
```

where `direction all` indicates that the host-name change needs to be made on both the target and source halves of the partition definition.

Example 4 (Alter Partition)

In the following example, `alter partition` is used to fix a partition definition that became invalid when the source application name (`oldAppName`) changed to `newAppName`, and affected only one half of the partition definition:

```
alter transparent partition newAppName.source to app2.target
    set application as newAppName instead of oldAppName direction single;
```

where `direction single` indicates that only one half of the partition definition needs to be corrected.

Note: The old application name can be discovered by issuing the `display partition` statement prior to correcting the partition definition.

Example 5 (Alter Partition)

In the following example, `alter partition` is used to fix a partition definition that became invalid when the source application name (`oldAppName`) changed to `newAppName`, and affected both halves of the partition definition because both partitioned databases were on the same application:

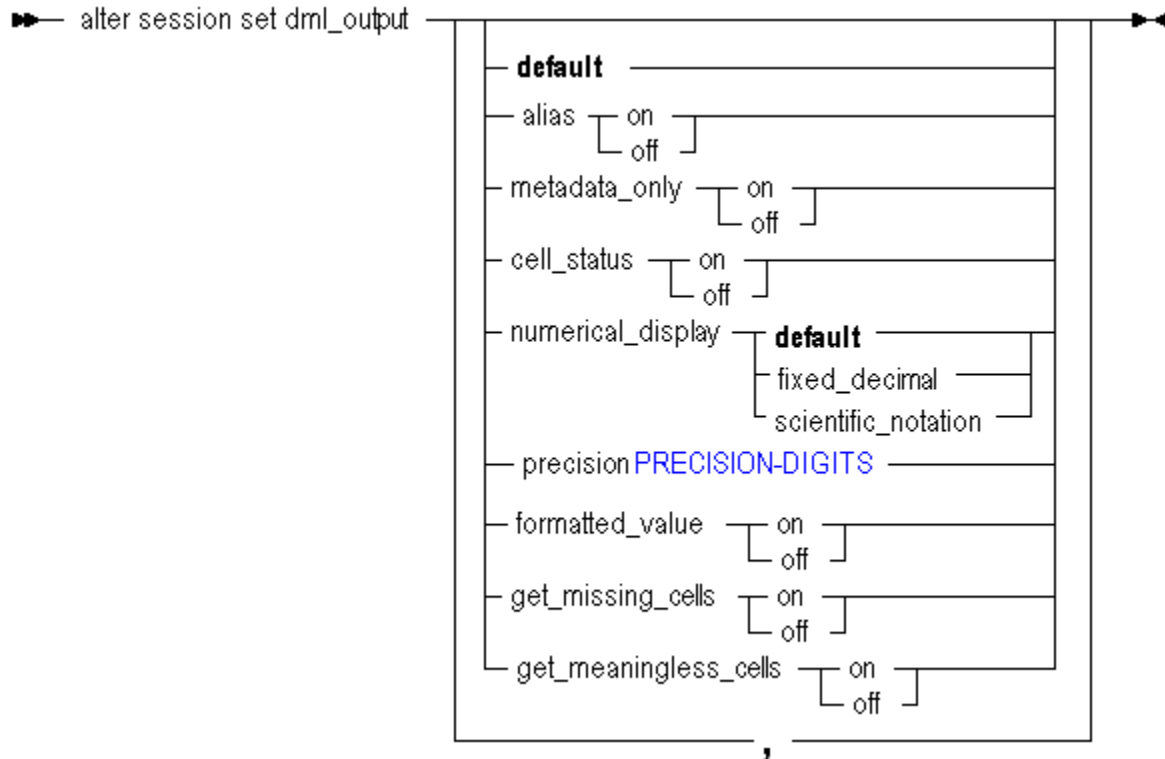
```
alter transparent partition newAppName.source to newAppName.target
    set application as newAppName instead of oldAppName direction all;
```

where `direction single` indicates both halves of the partition definition need to be corrected.

Alter Session

Set MDX display options.

Syntax



Use `alter session` to change the following MDX output settings:

Keyword	Description
<code>default</code>	Revert to the default MDX display settings in the MaxL Shell. The default settings are: <code>alias</code> ON, <code>metadata_only</code> OFF, <code>cell_status</code> OFF.
<code>alias on off</code>	Set whether to use aliases instead of member names.
<code>metadata_only on off</code>	Set whether to show only the metadata, with no data.
<code>cell_status on off</code>	Set whether to display cell status. Cell status is additional information returned with each cell value in MDX query outputs.

Note: Every cell consists of one member from each dimension. Up to four cell-status types may be returned with the output:

- **DC:** Dynamic Calc. If any of the members defining the cell is Dynamic Calc, this status is on.
- **RO:** Read Only. If the cell cannot be written to (for example, by lock-and-send), this status is on. Security filters in the database might cause cells to be read-only. Dynamic Calc cells are automatically read-only.
- **CM:** Calculated Member. If any of the members defining the cell is a calculated member, this status is on.
- **LO:** Linked Object. If the cell has any associated Linked Reporting Objects, this status is on.

Keyword	Description
numerical_display fixed_decimal scientific_notation default	Set whether MaxL returns data values in MDX query output as fixed decimals, scientific notation, or default format (values are returned in a reasonable combination of decimals or scientific notation).
precision <precision- digits>	Set the number (0-15) of decimal places to include for the data values in MDX query output.
formatted_value on off	Set whether to return formatted values for all cells of type text or date, or cells associated with a format string. By default, this setting is on.
get_missing_cells on off	Set whether to return #Missing valued cells for all cells of type text or date, or cells associated with a format string. By default, this setting is on.
get_meaningless_cells on off	<p>Set whether to return #Meaningless for cells that are empty only because they are unassociated with the context attribute or varying attribute. By default, this setting is off, and the empty cells display as #Missing.</p> <p>The following example query gets sales for all products, but the aggregation is specified by the slicer context only for Ounces_12.</p> <pre> SELECT {Sales, Cogs} ON COLUMNS, {Product.Levels(0).Members} ON ROWS FROM Sample.Basic WHERE (Ounces_12) ; </pre> <p>A value of #Meaningless is displayed for any members not associated with the attribute Ounces_12.</p>

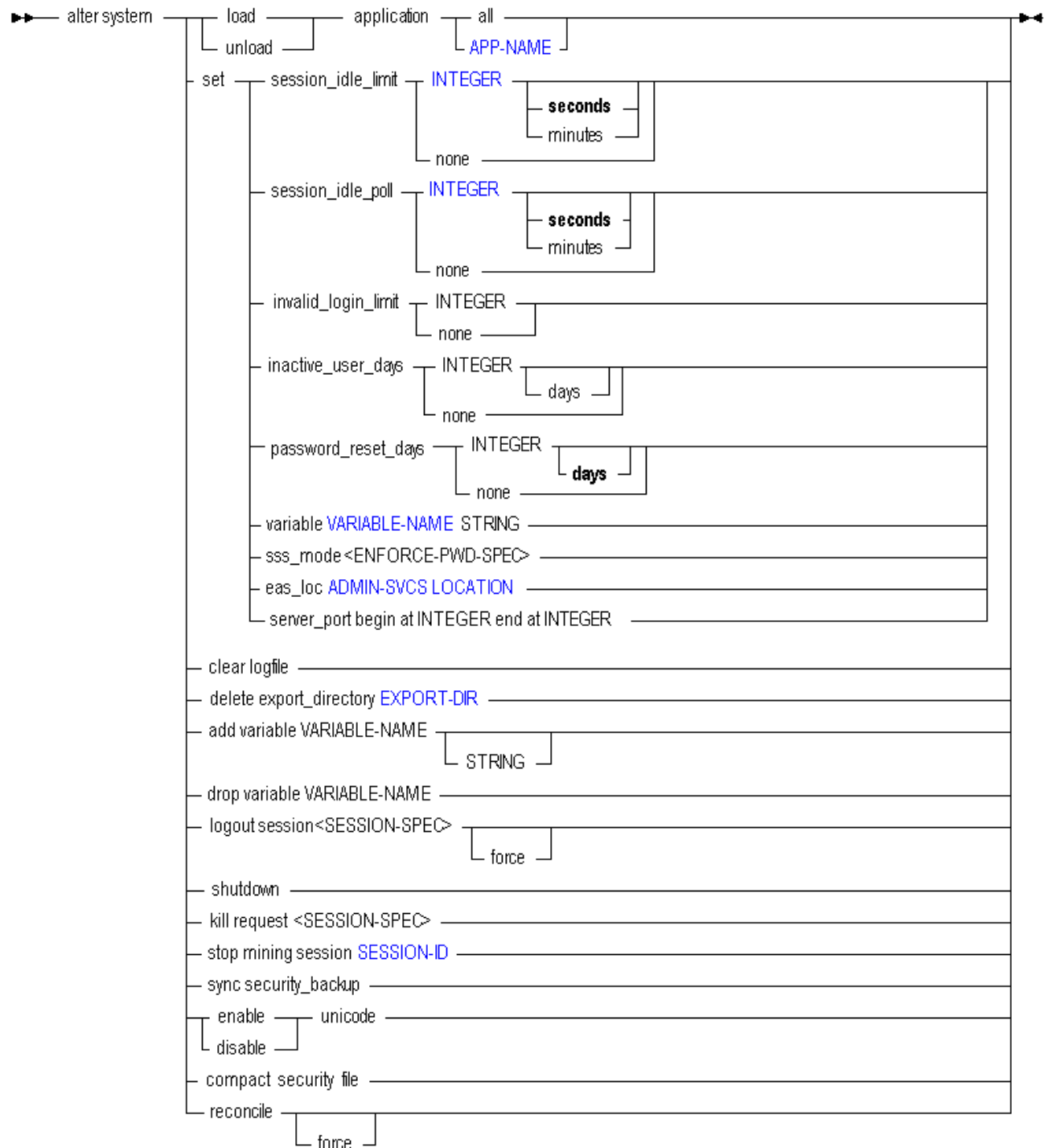
Alter System

[Click here for aggregate storage version](#)

Change the state of the Essbase Server. Start and stop applications, delete application log files, manipulate system-wide variables, manage password and login activity, disconnect users, end processes, back up the security file, and shut down the server.

Permission required: administrator.

Syntax



Use **alter system** to change the following system-wide settings:

Keyword	Description
load application	Start an application, or start all applications on the Essbase Server.
unload application	Stop an application, or stop all applications on the Essbase Server.

Keyword	Description
set session_idle_limit	<p>Set the interval of time permitted for a session to be inactive before Essbase Server logs off the user. The minimum limit that you can set is five minutes (or 300 seconds). When the session idle limit is set to none, all users can stay logged on until the Essbase Server is shut down.</p> <p>The default user idle logout time is 60 minutes. When a user initiates a calculation in the background, after 60 minutes the user is considered idle and is logged out, but the calculation continues in the background.</p> <p>Because the user may mistakenly assume that the calculation stopped because he or she was logged out, you can do one of the following to correct the user experience:</p> <ul style="list-style-type: none"> ● Run the calculation in the foreground ● Increase the session idle limit in to a time that exceeds the duration of the calculation, or to none
set session_idle_poll	<p>Set the time interval for inactivity checking and security-backup refreshing. The time interval specified in the session idle poll gives Essbase instructions:</p> <ul style="list-style-type: none"> ● Tells it how often to check whether user sessions have passed the allowed inactivity interval indicated by <code>session_idle_limit</code> in the alter system statement. ● Tells it how often to refresh the security backup file. If <code>session_idle_poll</code> is set to zero, the security backup file is still refreshed every five minutes.
set invalid_login_limit	<p>Set the number of unsuccessful login attempts allowed by any user before the system disables it. When you change this setting, the counter resets to 0. When the invalid login limit is set to none, there is no limit. By default, there is no limit.</p>
set inactive_user_days	<p>Set the number of days a user account may remain inactive before being disabled by the system. The counter resets when the user logs in, is edited, or is activated by an administrator. When the inactive days limit is set to none, user accounts remain enabled even if they are not used. By default, there is no limit.</p>
set password_reset_days	<p>Set the number of days users may retain passwords. After the allotted number of days, users are prompted at login to change their passwords. The counter resets for a user when the user changes the password, is edited, or is activated by an administrator. When the password reset days limit is set to none, there is no built-in limit for password retention. By default, there is no limit.</p>
set variable	<p>Change the value of an existing substitution variable on the system. The value must not exceed 256 bytes. It may contain any character except a leading ampersand (&).</p>

Keyword	Description
set sss_mode	<p>Migrate Essbase Server and any existing users and groups to Shared Services security mode. Minimum permission required: Administrator. After you have converted to Shared Services security mode, you cannot revert to native security mode.</p> <p>Password Enforcement Grammar:</p> <ul style="list-style-type: none"> ● enforce username_as_password—Create passwords that are the same as user names for users being migrated to Shared Services. <p>Note: The passwords are created in lowercase letters, even if the user name includes uppercase letters. For example, if a user name KSmith is migrated with this option, the password is ksmith.</p> <ul style="list-style-type: none"> ● enforce auto_password—Automatically generate new passwords for users being migrated to Shared Services. To see the generated passwords, use <code>display user all in shared_services_native</code> with <code>auto_password</code>; <p>Optionally save the generated passwords to a nondefault file location. If specifying a file name that already exists, use the <code>force</code> keyword to overwrite the file.</p> <p>If file name and location are not specified, the passwords are saved by default to <code>\$ARBORPATH\bin\MigratedUsersPassword.txt</code>.</p> <ul style="list-style-type: none"> ● enforce password <PASSWORD>—Generate the specified password for users being migrated to Shared Services.
set eas_loc	Set or change the Essbase Administration Server location that will be registered with Shared Services upon application creation or migration.
set server_port	<p>Expand a port range specified in <code>essbase.cfg</code>. Each Essbase application uses two ports from this range. If no more ports are available, an error message is displayed.</p> <p>Note: You can expand port ranges only so that the beginning port range is less than <code>SERVERPORTBEGIN</code> and the ending port range is greater than <code>SERVERPORTEND</code>.</p>
clear logfile	Clear accumulated entries from the Essbase Server log located in the <code>Essbase</code> directory. New log entries are created to record subsequent activity.
delete export_directory	<p>Delete directories created for linked reporting objects exported from a database to a directory created in <code>ARBORPATH\app</code>. Use this grammar after the exported LROs are migrated into a database using <code>import lro</code>, and the directories containing the exported LRO information are not needed.</p> <p>Note: This process works only for directories created in <code>ARBORPATH\app</code> using the <code>DBS-EXPORT-DIR</code> option of the <code>export lro</code> statement. It does not work for directories created elsewhere using the <code>FULL-EXPORT-DIR</code> option of the <code>export lro</code> statement.</p> <p>To view a list of names of exported linked-reporting-objects directories in <code>ARBORPATH\app</code>, use <code>display system export_directory</code>.</p>

Keyword	Description
add variable	Create a system-level substitution variable by name, and optionally assign a string value for the variable to represent. You can assign or change the value later using set variable . A substitution variable acts as a global placeholder for information that changes regularly. Substitution variables may be referenced by calculations and report scripts. If substitution variables with the same name exist at server, application, and database levels, the order of precedence for the variables is as follows: a database-level substitution variable supersedes an application-level variable, which supersedes a server-level variable.
drop variable	Remove a substitution variable and its corresponding value from the system.
logout session all	Terminate all user sessions currently running on the Essbase Server.
logout session...force	Terminate a session (or sessions) even if it is currently processing a request. The request is allowed to proceed to a safe point, and then the transaction is rolled back.
logout session <session-id>	Terminate a session by its unique session ID number. To see the session ID number, use display session .
logout session by user	Terminate all current sessions by a particular user, either across the entire Essbase Server, or limited to a specific application or database.
logout session by user on application	Terminate all current sessions by a particular user across a specific application.
logout session by user on database	Terminate all current sessions by a particular user across a specific database.
logout session on application	Terminate all current user sessions across a specific application.
logout session on database	Terminate all current user sessions across a specific database.
shutdown	Shut down the Essbase Server.
kill request all	Terminate all current requests on the Essbase Server.
kill request <session- id>	Terminate the current request indicated by the session ID. You can obtain session IDs using display session .
kill request by user	Terminate all current requests by the specified user on the Essbase Server.
kill request on application	Terminate all current requests on the specified application.
kill request on database	Terminate all current requests on the specified database.
stop mining session	Terminate the current data-mining session. The session ID of the data-mining session can be determined using display mining session .

Keyword	Description
sync security_backup	<p>Check whether the security backup file is the same as the security file, and if not, synchronize the security backup file with the current state of Essbase security. The effect is to refresh the backup file with any additions, changes, or deletions related to applications, databases, users, groups, filters, permissions, substitution variables, locked objects, and system settings.</p> <p>If <code>sync security_backup</code> is not issued directly as described above, the security backup file is checked/refreshed automatically at the same frequency with which session inactivity is checked globally. The default inactivity check interval is five minutes. To change the interval, use <code>set session_idle_poll</code>, or see the <i>Oracle Essbase Administration Services Online Help</i>.</p>
enable unicode	Set the Essbase Server to allow the creation of Unicode-mode applications and the migration of non-Unicode-mode applications to Unicode-mode applications.
disable unicode	Prevent the Essbase Server from allowing the creation of Unicode-mode applications or the migration of non-Unicode-mode applications to Unicode-mode applications.
compact security file	Defragment the security file. Fragmentation can gradually develop when objects such as users, groups, applications or databases are removed or changed. Please note that this operation slows down agent activity until the operation is completed, which could take a few minutes.
rename global registration name	Change the name of the global application and application project in Shared Services.
reconcile	<p>When Essbase is started using a security backup file (<code>essbase_timestamp.bak</code>) instead of <code>essbase.sec</code>, reconcile the security file to match the state of Essbase on an external disk. This grammar displays discrepancies in application and database information between the security file and the external disk:</p> <ul style="list-style-type: none"> ● If an application folder is on the disk but not in the security file, display a message indicating the discrepancy. (Essbase checks for the presence of a <code>appname/appname.app</code> file in the <code>ARBORPATH/app</code> directory.) The <code>force</code> option does not apply in this scenario. ● If an application file is in the security file but not on the disk, display a message indicating the discrepancy. The <code>force</code> option removes the application from the security file. ● If an application database folder is on the disk but not in the security file, display a message indicating the discrepancy. (Essbase checks for the presence of a <code>dbname/dbname.otl</code> file in the <code>ARBORPATH/app/appname</code> directory.) The <code>force</code> option does not apply in this scenario. ● If an application database file is in the security file but not on the disk, display a message indicating the discrepancy. The <code>force</code> option removes the database from the security file.

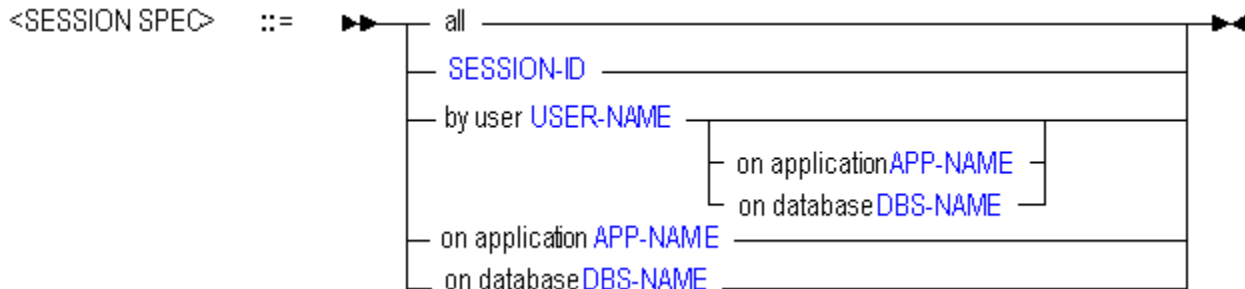
Notes

SESSION SPECIFICATION

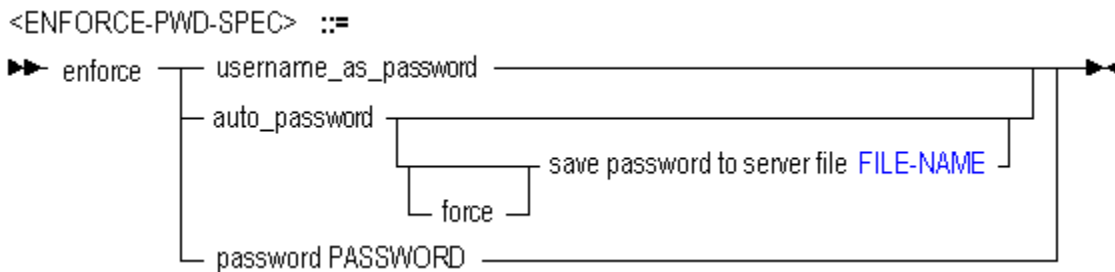
A *session* is a single user connection to Essbase Server. The session can be identified by keywords and names indicating context, or by a unique session ID number.

A *request* is a query sent to Essbase Server by a user or by another process; for example, starting an application or restructuring a database outline. Only one request at a time can be processed in each session.

If a session is processing a request at the time that an administrator attempts to terminate the session, the administrator must either terminate the request first, or use the **force** keyword available with **alter system** to terminate the session *and* the current request.



PASSWORD ENFORCEMENT SPECIFICATION



Example

```
alter system unload application Sample;
```

Stops the Sample application, if it is currently running.

```
alter system logout session by user Fiona;
```

Disconnects Fiona from any applications or databases to which she is connected.

Note: To log out a user, log out the sessions owned by that user.

```
alter system set password_reset_days 10;
```

Specifies that all users will be prompted after 10 days to change their passwords. The day count for any user is reset when the user changes the password or is edited or reactivated by an administrator.

```
alter system set sss_mode enforce password "password";
```

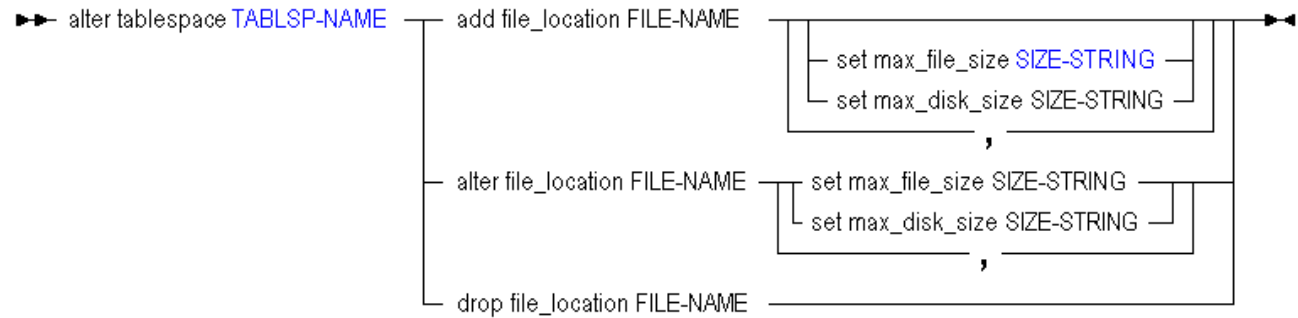
Migrates the Essbase Server to Shared Services security mode, specifying the initial password for all users.

Alter Tablespace

Change details about a tablespace. Tablespaces are applicable only to aggregate storage databases. To see a list of tablespaces, use [display tablespace](#). You cannot change the location or size of the metadata and log tablespaces.

Permission required: Application Manager. This statement requires the application to be started.

Syntax



Use `alter tablespace` to edit tablespaces in the following ways:

Keyword	Description
add file_location	<p>Add a new file location to the tablespace.</p> <p>Note: FILE-NAME is case sensitive in this statement.</p>
alter file_location	<p>Change the maximum file-size or disk-size value for the specified file location.</p> <p>Note: FILE-NAME is case sensitive in this statement.</p>
set max_file_size	<p>Specify a value for the maximum size that a data file may attain before Essbase creates a new file.</p> <p>The largest possible value that the aggregate storage kernel can handle is 134217727 MB. This is also the default value. If operating system limits take effect before this value is reached, the kernel creates a new file. If you enter a value that is larger than 134217727 MB, the kernel ignores the setting and caps file size at 134217727 MB.</p> <p>The minimum value is 8MB (8388608b), and any values you enter are rounded up to the next 8MB interval.</p>
set max_disk_size	<p>Specify the value for the maximum amount of disk space to be allocated to the file location.</p> <p>The largest possible value that the aggregate storage kernel can handle is 4294967295 MB. This is also the default value. If operating system limits take effect before this value is reached, the kernel attempts to use another file location in the tablespace. If you enter a value that is larger than 4294967295 MB, the kernel ignores the setting and caps disk size at 4294967295 MB.</p> <p>The minimum value is 8MB (8388608b), and any values you enter are rounded up to the next 8MB interval.</p>
drop file_location	<p>Delete the specified file location from the tablespace. When a file location is deleted, all files in the file location are deleted, as well as the subdirectory containing the files. You cannot delete a file location if it contains data. You cannot delete the tablespace itself.</p> <p>Note: FILE-NAME is case sensitive in this statement.</p>

Example

```
alter tablespace asosamp.'default' add file_location 'C:\\mytablespace' set
max_file_size 50mb;
```

Adds another file location for the default tablespace. Now the tablespace default is in C:\mytablespace in addition to the original location, C:\Hyperion\products\Essbase\EssbaseServer\app.

```
alter tablespace asosamp.'default' alter file_location 'C:\\Hyperion\\products\\Essbase\\
\EssbaseServer\\' set max_file_size 50mb;
```

Changes the maximum file size allowed in the specified location of the default tablespace. Note that the file_location string is case sensitive.

Alter Trigger

Enable or disable a trigger created to track state changes over a selected cube area.

For more information about the Essbase triggers feature, see the *Oracle Essbase Database Administrator's Guide*.

Syntax

```
▶▶ alter trigger TRIGGER-NAME enable
      TRIGGER-NAME disable
      on database DBS-NAME disable
```

Use **alter trigger** to edit triggers in the following ways:

Keyword	Description
enable	Essbase monitors the trigger during data load, calculation or lock and send. Essbase performs the trigger action when the specified condition is met on the specified cube area.
disable	Essbase does not monitor the trigger.
on database <DBS-NAME> disable	Essbase disables all triggers currently enabled in the database. A restart of the application or the database following the disable restores the triggers to the same state as before the disable was issued (all the triggers disabled using <code>alter trigger on database DBS-NAME disable</code> are re-enabled).

Example

```
alter trigger Sample.Basic.WatchCosts disable;
alter trigger on database sample.basic disable;
```

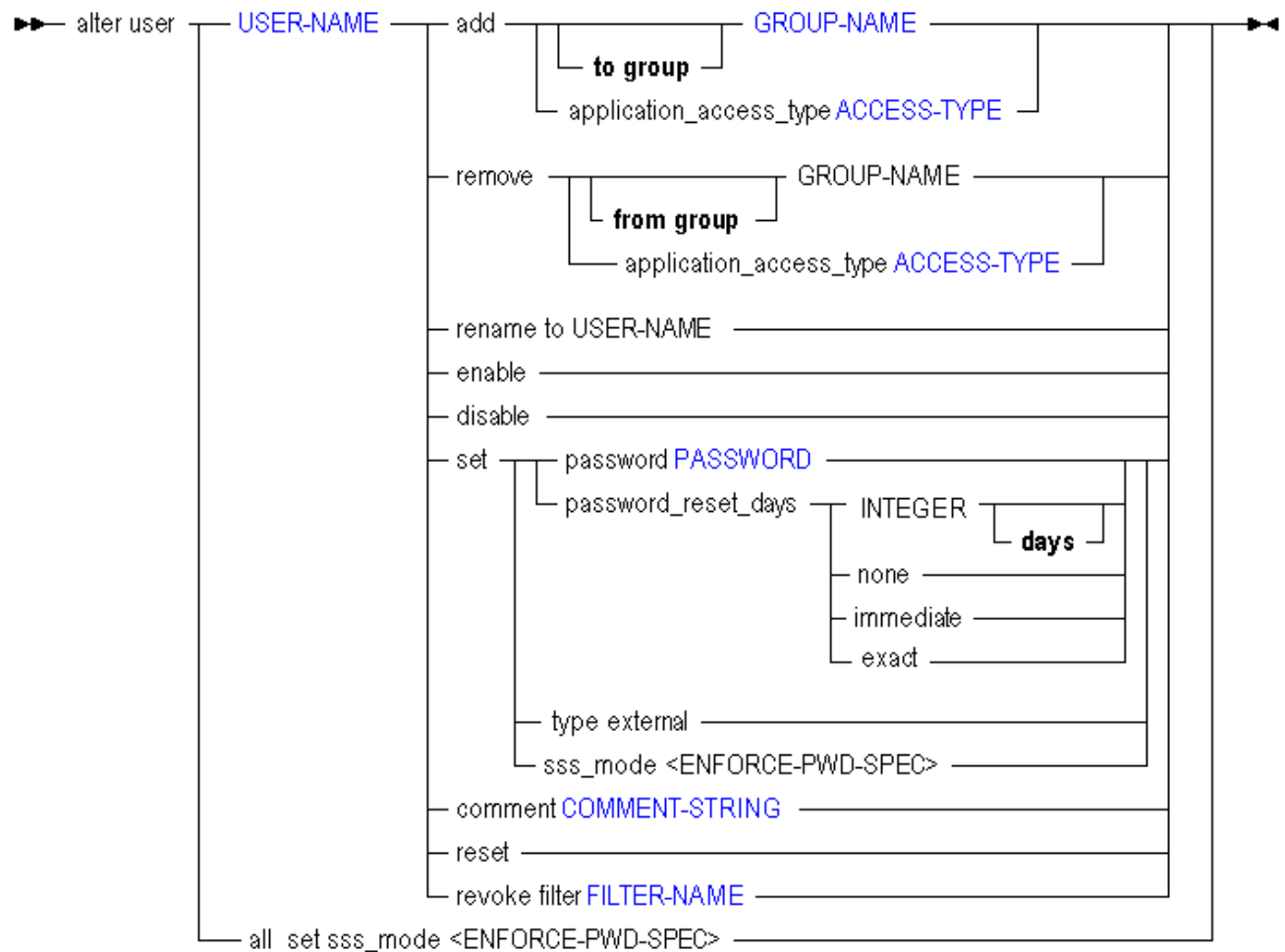
Alter User

Add or remove a user to or from a group. Rename a user. Change the comment that describes a user. Enable or disable a user account. Change a user's password, or specify whether it should expire. Control user application access to application domains.

Permission required: create_user.

When Essbase runs in EPM System security mode, the Essbase create_user permission becomes obsolete. You must be an Essbase administrator to manage users, and you must additionally be a Shared Services administrator to manage users from Shared Services.

Syntax



Use **alter user** to change user information in the following ways:

Keyword	Description
add [to group]	Add the user to a group. In EPM System security mode, this action automatically causes a user/group synchronization between Essbase and Shared Services. It is advisable to use Shared Services to manage users and groups instead.
add application_access_type	Add an application access type. An application access type controls which domains a user can access based on the named user license. To view a list of the user's allowed application access types, use display user . MaxL can be used only to add or remove Essbase access.

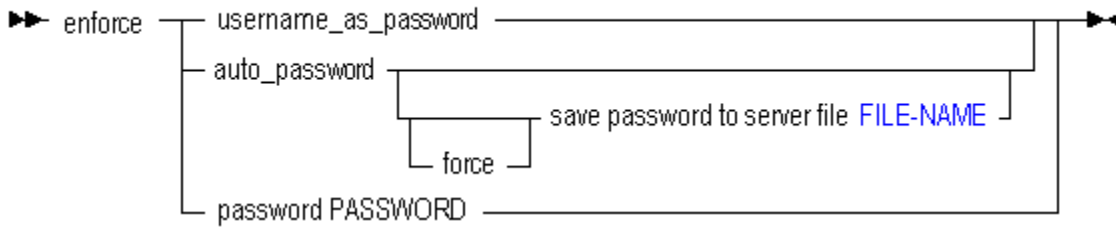
Keyword	Description
remove [from group]	<p>Remove the user from a group.</p> <p>In EPM System security mode, this action automatically causes a user/group synchronization between Essbase and Shared Services. It is advisable to use Shared Services to manage users and groups instead.</p> <p>Note: If you deprovision a group in Shared Services that was provisioned to Essbase, users in the group remain in the Essbase security file after performing a single group synchronization. To deprovision a group and its users, perform a full system refresh.</p>
remove application_access_type	Remove an application access type. MaxL can be used only to add or remove Essbase access.
rename to	Rename the user.
enable	Reactivate the user if the user's permission to log in has been terminated.
disable	Disable the user's permission to log in to Essbase.
set password	Change the user's password.
set password_reset_days INTEGER days	Specify the number of days before a password expires. This setting has meaning only if the system-level password_reset_days value (shown in the password_reset_days field of display system) is not zero or "none". The value of this setting must be between 1 and 65535. The latest effective date for user-level password expiration is Jan 19, 2038.
set password_reset_days none	Remove any user-level password expiration setting created by alter user set password_reset_days INTEGER , and revert the password reset days value back to the system-level value (shown in the password_reset_days field of display system).
set password_reset_days immediate	Force the user to change password at the next login.
set password_reset_days exact	Undo the 'immediate' setting above. If the administrator chooses 'immediate' and then attempts to revert to allowing a set number of days, the setting will not work, because 'immediate' takes precedence. Using 'exact' is the only way to reverse 'immediate.'
set type external	Specify that this user must log in to Essbase using Shared Services. For the user to log in successfully, the "AUTHENTICATIONMODULE" on page 395 parameter must be set to CSS in the <code>essbase.cfg</code> file, and the user name must match a valid user name in the external authentication repository.

Keyword	Description
set sss_mode	<p>Migrate the user to EPM System security mode. This action might be useful if the user migration failed using <code>alter system</code>. Minimum permission required: Administrator.</p> <p>Password Enforcement Grammar:</p> <ul style="list-style-type: none"> ● enforce username_as_password—Create passwords that are the same as user names for users being migrated to Shared Services. <p>Note: The passwords are created as lowercase, even if the user name contains uppercase letters. For example, if a user name <code>KSmith</code> is migrated with this option, the password will be <code>ksmith</code>.</p> <ul style="list-style-type: none"> ● enforce auto_password—Automatically generate passwords for the users being migrated to Shared Services. To see the generated passwords, use <code>display user all in shared_services_native with auto_password</code>; <p>Optionally save the generated passwords to a nondefault file location. If specifying a file name that exists, use the <code>force</code> keyword to overwrite the file.</p> <p>If file name and location are not specified, passwords are saved by default to <code>\$ARBORPATH\bin\MigratedUsersPassword.txt</code>.</p> <ul style="list-style-type: none"> ● enforce password <PASSWORD>—Generate the specified password for users being migrated to Shared Services. <p>For more information, see the <i>Oracle Essbase Database Administrator's Guide</i> chapter titled "User Management and Security."</p>
comment	Create a description of the user.
reset	<p>Remove obstructions to logging in for the specified user account.</p> <ul style="list-style-type: none"> ● The user account is re-enabled if it was disabled. ● Any requirement to change password immediately is removed. ● If the password has expired, the expiration is cleared. ● The count of unsuccessful user logins is reset to 0.
revoke filter	<p>Remove a filter assignment to this user. Privilege required: Application manger.</p> <p>Note: This statement does not remove filter assignments gained by membership to groups. To remove filter assignments to groups, use <code>Alter Group</code>.</p>
all set sss_mode	Same as <code>set sss_mode</code> , but for all users.

Notes

PASSWORD ENFORCEMENT SPECIFICATION

<ENFORCE-PWD-SPEC> ::=



Example

```
alter user Fiona add to group Newhires;
```

Assigns Fiona to a group called Newhires.

```
alter user Fiona enable;
```

Enables user Fiona to log in again.

```
alter user Fiona set password_reset_days immediate;
```

Requires Fiona to change password at the next login.

```
alter user 'Autumn Smith' set type external;
```

Specifies that Autumn Smith is externally authenticated in a supported authentication repository (LDAP, Microsoft Active Directory, or Windows NT LAN Manager).

```
alter user ASmith rename to 'Autumn Smith';  
alter user 'Autumn Smith' set type external;
```

Renames native Essbase user Asmith to Autumn Smith, because that is the name stored in the authentication repository. Specifies that Autumn Smith is externally authenticated in a supported authentication repository.

```
alter user Fiona remove application_access_type Essbase;
```

Removes Essbase application access from user Fiona. If user Fiona has permission to access Oracle Hyperion Planning, Fusion Edition, that permission remains intact.

Create Application

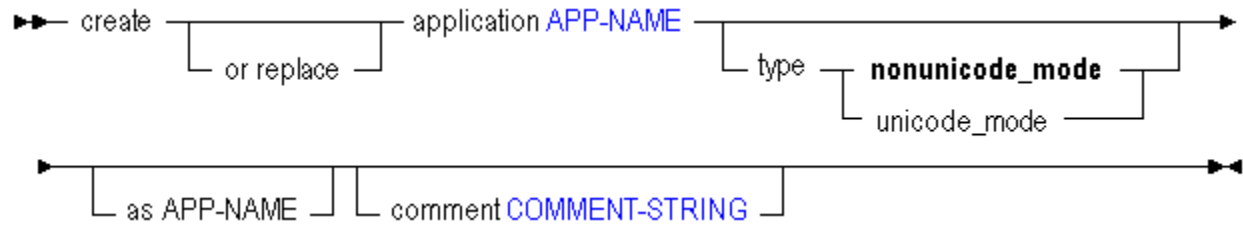
[Click here for aggregate storage version](#)

Create or re-create an application, either from scratch or as a copy of another application on the same system. APP-NAME must consist of 8 or fewer characters. Avoid spaces and special characters when naming applications and databases. Application names are not case-sensitive.

Permission required: Essbase create_application role and Shared Services Project Manager role.

To copy an application, Manager permission on the source application is also required.

Syntax



Use `create application` to create an application in the following ways:

Keyword	Description
<code>create application</code>	Create a new application. Application names are not case-sensitive.
<code>create or replace application</code>	Create an application, or replace an existing application of the same name. Application names are not case-sensitive.
<code>...type nonunicode_mode</code>	Create a Non Unicode-mode application. This is also the default if these keywords are omitted.
<code>...type unicode_mode</code>	Create a Unicode-mode application.
<code>create application as</code>	Create an application as a copy of another application. Application names are not case-sensitive.
<code>comment</code>	Create an application description (optional). The description can contain up to 80 characters.

Example

```
create application Sample comment 'This is a test application.';
```

Creates a new application called Sample with an associated comment.

```
create application Newsamp as Sample;
```

Creates an application called Newsamp which is a copy of the application Sample.

```
create or replace application Sample;
```

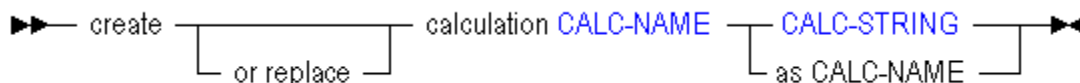
Creates an application called Sample. If an application named Sample already exists, it is overwritten.

Create Calculation

Create, replace, or copy a stored calculation.

Permissions required: Database Manager to create database-level calculations. Application Manager to create application-level calculations.

Syntax



Use **create calculation** to create a calculation in the following ways:

Keyword	Description
create calculation	Create a calculation script, the body of which is specified by “CALC-STRING” on page 776.
create or replace calculation	Create a calculation script, the body of which is specified by “CALC-STRING” on page 776. If a calculation script of that name already exists, it is replaced.
create calculation as	Create a calculation as a copy of another stored calculation.

Notes

- When creating database-level calculations, this statement requires the database to be started.
- A stored calculation can be associated with an application/database, or with an application only. To create an application-level calculation, use two tokens for CALC-NAME. To create a database-level calculation, use three tokens. See “CALC-NAME” on page 774 for more details.
- Calculations created using MaxL must be valid. For information about calculation syntax, see the *Oracle Essbase Database Administrator's Guide*.

Example

```
create or replace calculation sample.basic.Accts
'SET UPDATECALC ON;
CALC DIM(Accounts);'
;
```

Creates a calculation named Accts that is associated with sample.basic.

```
create calculation sample.basic.Accts2 as app.db.Accts
```

Creates a calculation named Accts2 on sample.basic that is a copy of another database's calculation named Accts.

Create Database

[Click here for aggregate storage version](#)

Create or re-create a regular or currency database. Optionally create the database as a copy of another database on the same system. DBS-NAME must consist of 8 or fewer characters. Avoid spaces and special characters when naming applications and databases.

Permission required: Application Manager. To copy a database, Manager permission on the source database is additionally required.

Syntax



Use **create database** to create a database in the following ways:

Keyword	Description
create database	Create a new database. Database names are not case-sensitive.
create or replace database	Create a database, or replace an existing database of the same name. Database names are not case-sensitive.
create database using non_unique_members	Create a database that supports the use of duplicate member names. Once you have created a database with a duplicate member outline, you cannot convert it back to a unique member outline. For more information about duplicate member names, see the <i>Oracle Essbase Database Administrator's Guide</i> chapter titled "Creating and Working With Duplicate Member Outlines."
create database as	Create a database as a copy of another database. Database names are not case-sensitive.
create currency database	Create or replace a database for currency conversion. Linking a currency database to a main database enables you to convert currency values in a database from one currency into another currency.
comment	Create a database description (optional). The description can contain up to 80 characters.

Example

```
create or replace database Sample.Basic comment 'This is a test.';
```

Creates a database called Basic within the Sample application. If a database named Basic within the Sample application already exists, it is overwritten.

```
create database Sample.New as Sample.Basic;
```

Creates a database called New within the Sample application that is a copy of the database Basic within the Sample application.

```
create currency database Sample.Intern1;
```

Creates a currency database called Intern1 within the Sample application.

Create Drillthrough

Create a drill-through URL within the active database outline.

For each drillable region of an Essbase database, you can enable drill-through access by means of a URL to Web content hosted on Oracle ERP and EPM applications.

Syntax

```

▶▶ create drillthrough URL-NAME from xml_file FILE-NAME ▶▶
▶ on { MEMBER-EXPRESSION } [level0 only] ▶▶

```

Use **create drillthrough** to create a drill-through URL definition in the following ways:

Keyword	Description
create	Create a drill-through URL as metadata.
drillthrough	The number of drill-through URLs per database is limited to 255.
from xml_file	<p>Indicate the path to the local URL XML file that defines the link information.</p> <p>The URL XML is created by the ERP or EPM application that deployed the Essbase database. The XML contains the drill-through URL display name and a URL enabling the hyperlink from a cell to a Web interface to occur.</p> <p>The following is a sample URL XML file:</p> <pre><?xml version="1.0" encoding="UTF-8"?> <foldercontents path="/"> <resource name="Assets Drill through GL" description="" type="application/x-hyperion-applicationbuilder-report"> <name xml:lang="fr">Rapport de ventes</name> <name xml:lang="es">Informe de ventas</name> <action name="Display HTML" description="Launch HTML display of Content" shortdesc="HTML"> <url>/fusionapp/Assetsdrill.jsp?\$\$SSO_TOKEN\$\$&\$\$CONTEXT\$\$& \$ATTR(ds,pos,gen,level.edge)\$ </url> </action> </resource> </foldercontents></pre>
on {<member-expression>,...}	<p>Define the list of drillable regions, using the same Essbase member-set calculation language that is used to define security filters. The list of drillable regions must be enclosed in {brackets}.</p> <p>The number of drillable regions in a drill-through URL is limited to 256. The number of characters per drillable region is limited to 65536.</p>
level0 only	Optional: Restrict the URL definition to level-0 data.

Example

```
create drillthrough sample.basic.myURL from xml_file "C:/drillthrough/data/myfile1.xml"
on {'@Ichildren("Qtr1")', '@Ichildren("Qtr2")'} level0 only;
```

See Also

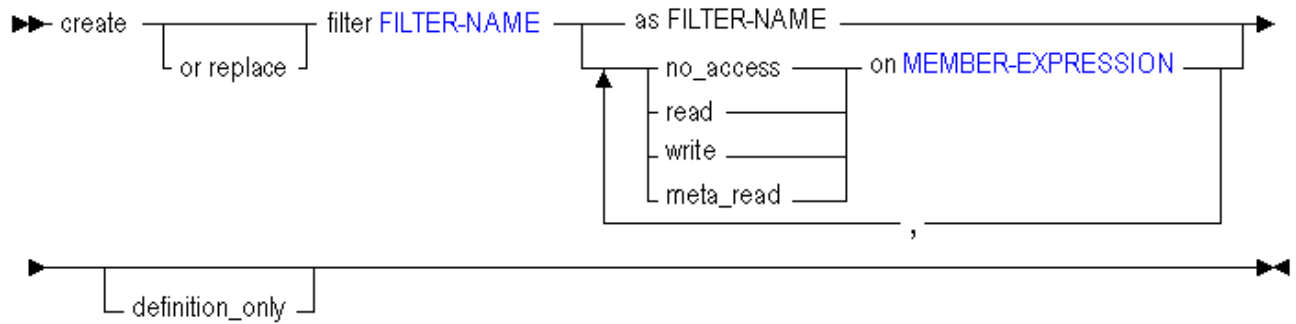
- [alter drillthrough](#)
- [display drillthrough](#)
- [drop drillthrough](#)

Create Filter

Create or re-create a database security filter, either from scratch or as a copy of another filter on the same system. Filters control security for database objects. Use [grant](#) to assign filters to users and groups.

Minimum permission required: Database Manager.

Syntax



Use **create filter** to create a filter in the following ways:

Keyword	Description
create filter	Create a security filter to restrict or permit access to specified database cells.
create or replace filter	Create a security filter or replace an existing security filter of the same name.
create filter ... no_access on <member-expression>	Create a filter blocking access to a specified member combination.
create filter ... read on <member-expression>	Create a filter providing read-only access to a specified member combination.
create filter ... write on <member-expression>	Create a filter providing write access to a specified member combination.
create filter ... meta_read on <member-expression>	Create a filter restricting access to siblings and ancestors of the member expression. In case of a filtering conflict, the MetaRead filtering overrides the other filter permissions. For more information about metadata filtering, see “Metadata Filtering” on page 926 .
create or replace filter ... definition_only;	Updates the filter definition while retaining user associations with the filter. If you replace a filter without using <code>definition_only</code> , then the filter must be re-granted to any users to whom it was assigned.

Notes

- Filters created using MaxL must be valid. For information about filter syntax, see the *Oracle Essbase Database Administrator's Guide*.
- MEMBER-EXPRESSION must be enclosed in single quotation marks. It can be a comma-separated list.

Example

```
create filter sample.basic.filt1 read on 'Jan, sales', no_access on '@CHILDREN(Qtr2)';
```

Creates a filter to restrict privileges to Sample Basic as follows: gives read-only access to the intersection of Jan and sales (sales data for January only); blocks access to children of Qtr2 (April, May, and June).

```
create or replace filter sample.basic.filt1 read on 'Sales, @ATTRIBUTE(Bottle)';
```

Creates a filter (or changes an existing filter) to restrict privileges to Sample Basic as follows: gives read-only access to sales data for products packaged in a bottle (product base dimension members associated with the Bottle attribute member).

Create Function

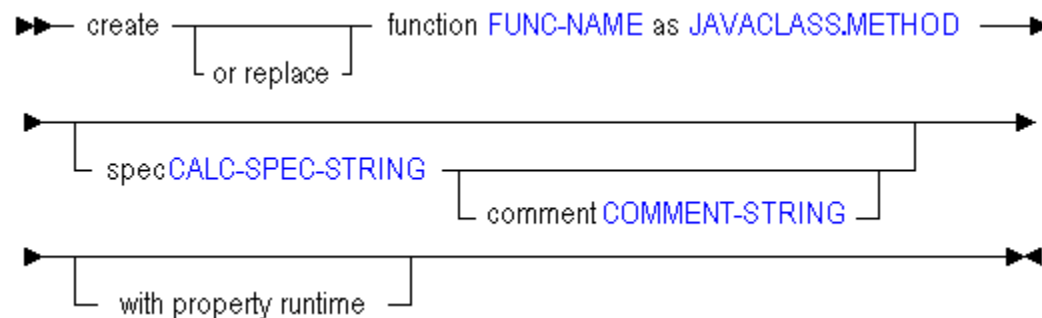
Create or re-create your own registered Essbase calculation function, using a Java method.

Minimum permission required: Application Manager to create a local (application-level) function. Administrator to create a global (system-level) function.

Process to follow:

1. Develop the functions in Java classes.
2. Use `create function` to register them in the Essbase calculator framework.
3. You can now use the functions in the same way that you use the standard Essbase calculation functions.

Syntax



Use `create function` to create a function in the following ways:

Keyword	Description
create function as	Register with Essbase a custom-defined function developed in Java, either as a global function usable by the entire Essbase Server, or as a local function available to an application. To register a global (server-wide) function, use one token for “ FUNC-NAME ” on page 786. To register a local (application-wide) function, use two tokens for “ FUNC-NAME ” on page 786.
create or replace function as	Register with Essbase a global or local custom-defined function. If a function with that name already exists in the custom-defined function and macro catalog, it is replaced.
spec	Enter, for the custom-defined function, an optional Essbase calculator-syntax specification string, such as in the following example: <code>@COVARIANCE (expList1, expList2)</code> . Use a specification string if you wish the function to be returned by the output string of the <code>EssListCalcFunctions</code> API function.

Note: If you do not specify a calculation specification string, you cannot specify a comment either.

Keyword	Description
with property runtime	Designate the custom-defined function as a runtime function. Normally, Essbase pre-executes functions whose arguments are available at compilation time. The Runtime property prevents that optimization, executing functions that have constant values as operands (or no operands at all) for every block in the function range. If the built-in @CALCMODE(CELL) function is used, a custom-defined function declared as Runtime can execute on every cell in the range.

Note: No built-in Essbase calculator functions have the Runtime property.

The Runtime property should be applied only in special circumstances, as it can seriously affect performance. The runtime property might be desirable for any custom-defined function whose return value depends on something besides its arguments; for example, the current date, or values in a rapidly changing relational table. If you created a runtime function @RANDOM() that returns a new random number each time it executes, then a member formula such as "Mem1 = @RANDOM () ; " would return different values for each block. At compilation time, the Runtime property prevents the pre-execution of functions that are applied to constants.

comment	Create a description of the function (optional). You cannot create a comment without also using spec to create a calculator-syntax specification string. The optional calculator-syntax specification string and the comment are used as the output string of the EssListCalcFunctions API function.
---------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Notes

- To create a global or system-level function, use a single name for FUNC-NAME. For example, '@COVARIANCE'.
- To create a local or application-level function, use MaxL's double naming convention for FUNC-NAME. For example, Sample.'@COVARIANCE'. The second token must be enclosed in single quotation marks because it contains a special character.

Example

```
CREATE FUNCTION '@COVARIANCE'
AS 'com.hyperion.essbase.calculator.Statistics.covariance'
SPEC '@COVARIANCE (expList1, expList2)'
COMMENT 'computes covariance of two sequences given as expression lists';
```

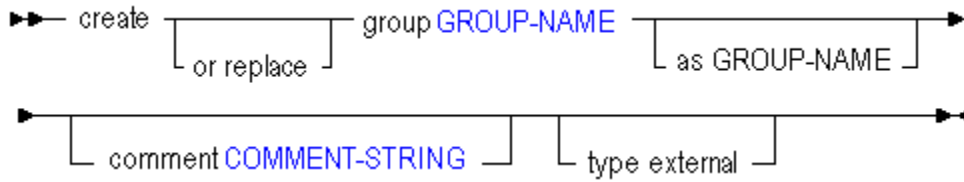
Create Group

Create or re-create a group, either from scratch or as a copy of another group.

An Essbase group can be created in two ways:

1. From Essbase using MaxL or Administration Services. To do this, you must be an Essbase administrator. If Essbase is running in Shared Services mode, you must additionally have the Shared Services Directory Manager role.
2. From Shared Services. To do this, you must be a Shared Services administrator.

Syntax



Use **create group** to create a group in the following ways:

Keyword	Description
create group	Create a security group to assign users to, so that they can share identical minimum permissions assigned at the group level.
create or replace group	Create a security group. If a group of that name already exists, it is replaced.
create group as	Create a group as a copy of an existing group.
comment	Create a description of the security group.
type external	For use only in EPM System security mode. Create and provision in Essbase a group that already exists in Shared Services.

Example

```
create group Level_1 as Newhires comment 'Copy of Newhires';
```

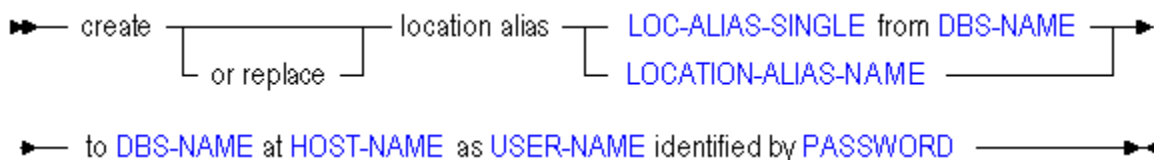
Creates a group called Level_1 that is a copy of the existing group Newhires.

Create Location Alias

Create on the database a location alias identifying a host name, database, user name, and password. Location aliases provide a shorthand way of referencing login information for other Essbase databases.

Minimum permission required: Database Manager.

Syntax



Use **create location alias** to create a location alias in the following ways:

Keyword	Description
create location alias	Create a location alias, identifying a remote host name, database, user name, and password. The location alias can be used by the @XREF function as an abbreviated login to a remote database.
create or replace location alias	Create a location alias, replacing any existing location alias of the same name on the same database.

Keyword	Description
...from <dbs-name>	Specify the name of the current database (the database on which the location alias is being created).
...to <dbs-name>	Specify the name of the remote database to log in to.
...at <host-name>	Specify the remote host name on which the remote database resides.
...as <user-name> identified by <password>	Specify a user name and password with which to log in to the remote database.

Notes

- This statement requires the database to be started.
- Location aliases created using MaxL must be valid. For information about location aliases, see the *Oracle Essbase Database Administrator's Guide*.
- Location aliases are used by the @XREF function for cross-database calculations.

Example

```
create location alias EasternDB from Sample.Basic to East.Sales at Easthost as Fiona
identified by sunflower;
```

Creates a location alias called EasternDB on Sample.Basic that represents the following login information:

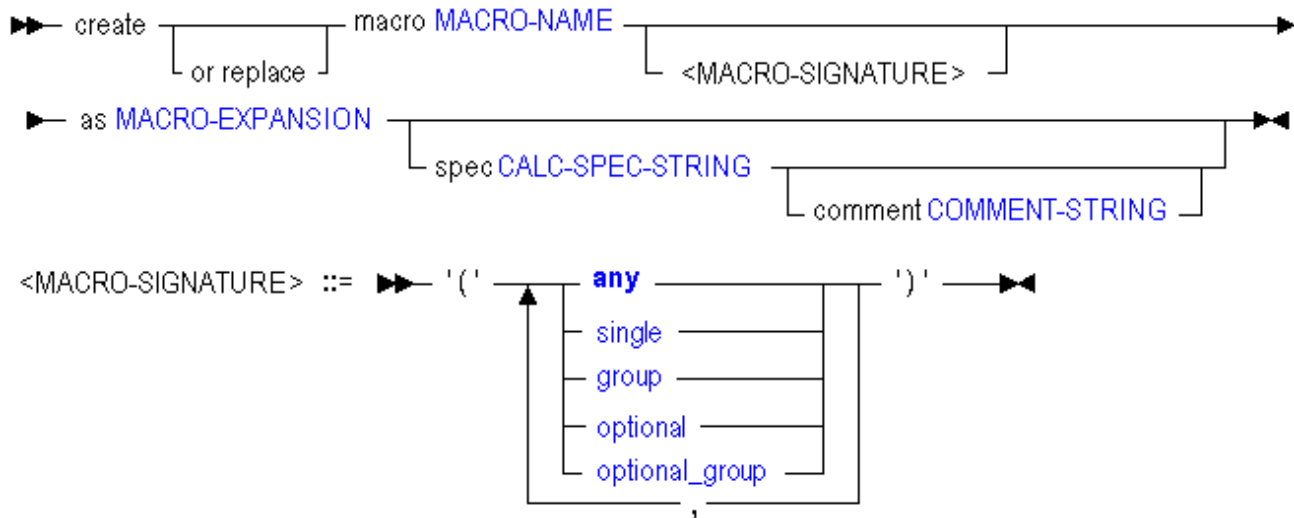
- server = Easthost
- application = East
- database = Sales
- user name = Fiona
- password = sunflower

Create Macro

Create or re-create your own Essbase calculation macro as your chosen combination of existing calculation functions or macros. This statement registers the new macro with the Essbase custom-defined function and macro catalog.

Minimum permission required: Application Manager to create a local (application-level) macro. Administrator to create a global (system-level) macro.

Syntax



Use `create macro` to create a macro in the following ways:

Keyword	Description
create macro as	Create and register with Essbase a custom-defined macro as your chosen combination of existing calculation functions or macros. Register the macro either as a global macro usable by the entire Essbase Server, or as a local macro available to an application. To register a global (server-wide) macro, use one token for “MACRO-NAME” on page 793. To register a local (application-wide) function, use two tokens for “MACRO-NAME” on page 793.
create macro... <macro- signature>	Enter for the macro an optional signature defining the syntax rules for macro arguments. A macro signature describes the style in which arguments (or input parameters) to the macro may be passed. One example of a macro signature is (SINGLE, SINGLE, GROUP), meaning that the macro must be passed two comma-separated arguments followed by a list of arguments. For more information, see “Custom-Defined Macro Input Parameters” on page 292.
create or replace macro	Register with Essbase a global or local custom-defined macro. If a macro with that name already exists in the custom-defined function and macro catalog, it is replaced.
spec	Enter for the macro an optional calculator-syntax specification string, as in the following example: @MYMACRO (mbrName, rangeList). Use a specification string if you wish the macro to be returned by the output string of the EssListCalcFunctions API function.
	Note: If you do not specify a calculation specification string, you cannot specify a comment either.
comment	Create a description of the macro (optional). You cannot create a comment without also using <code>spec</code> to create a calculator-syntax specification string. The optional calculator-syntax specification string and the comment are used as the output string of the EssListCalcFunctions API function.

Notes

- To create a global (system-level) macro, use a single name for MACRO-NAME. For example, '@COVARIANCE'.
- To create a local (application-level) macro, use MaxL's double naming convention for MACRO-NAME. For example, Sample.'@COVARIANCE'.

Example

```
create macro Sample.'@COVARIANCE'(single, single) as '@COUNT(SKIPMISSING,@RANGE(@@S))'  
spec '@COVARIANCE (expList1, expList2)' comment 'Computes covariance of two sequences  
given as expression lists';
```

Create Partition

Create or validate a partition definition between two databases. Permission required: Database Manager at both sites.

Select the type of partition to create:

- [transparent](#)
- [replicated](#)
- [linked](#)

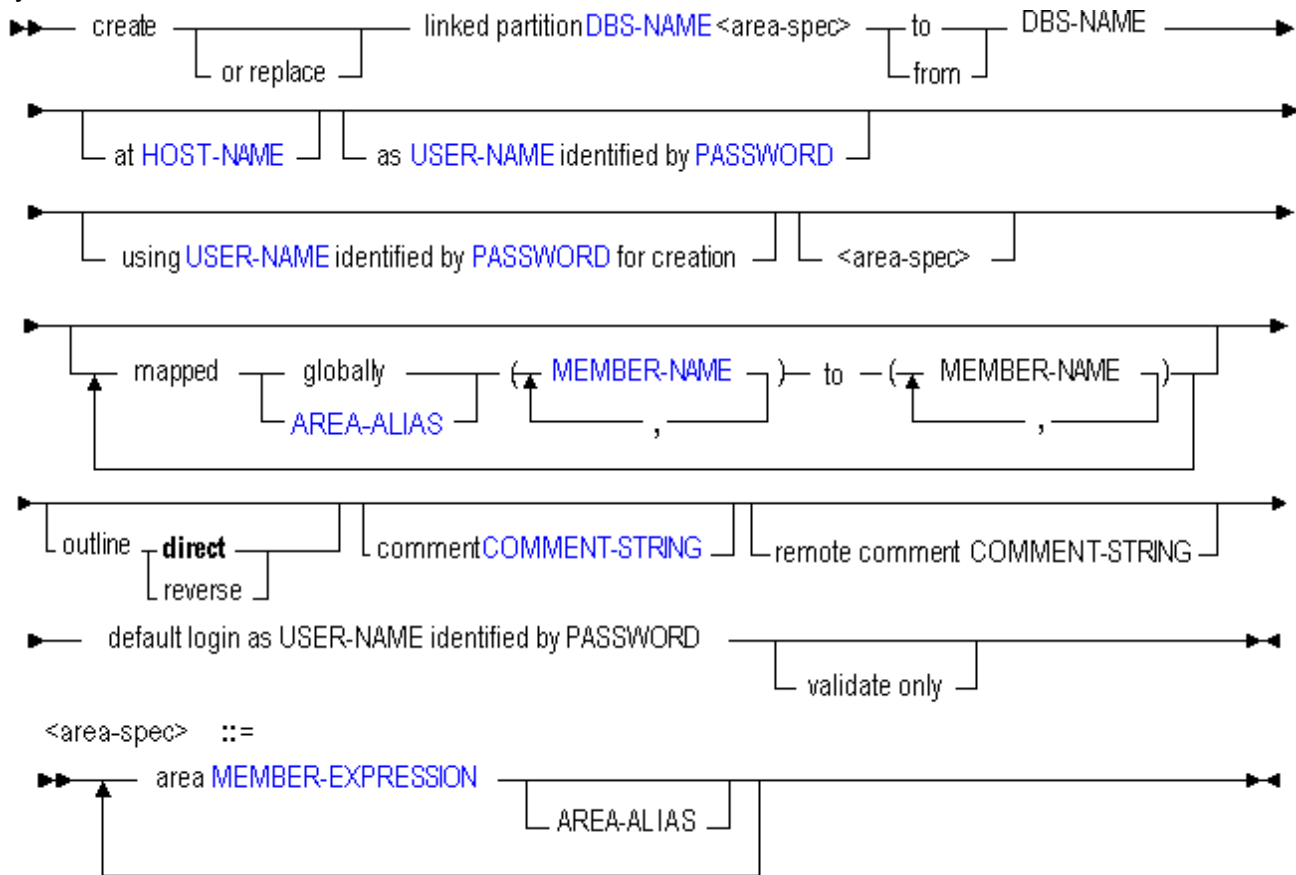
Partitions created using MaxL must be valid. To validate a partition, use the **validate only** clause. For information about partition definitions, see the *Oracle Essbase Database Administrator's Guide*.

Create Linked Partition

Create or validate a linked partition definition between two databases. A linked partition enables users to navigate from one data value in one database, to a subset of another database. The two databases may contain very different outlines.

For example, if a Spreadsheet Add-in in user clicks a database cell that contains a link to another database, a new sheet opens displaying the dimensions in the second database. The user can then drill down into the linked database's dimensions.

Syntax



Use **create linked partition** to create a partition in the following ways:

Keyword	Description
create linked partition	Create a linked partition. A linked partition connects two different databases with a data cell. The databases can contain largely different dimensions, and still be connected by a small, mapped data region. With linked partitions, the spreadsheet that a user first views is connected to the target, and the spreadsheet that opens when the user drills across is connected to the source.
create or replace ...partition	Create a partition definition, or replace an existing partition definition.
area...	Define the partition areas to share with the other database. Optionally nickname the area using an area-alias .
to <dbs-name>	Create a partition definition between the current database source and the second database (the target).
from <dbs-name>	Create a partition definition between the current database target and the second database (the source).
at <host-name>	Specify the remote computer name, if you are creating a partition definition between the current database and one residing on a remote Essbase Server host.

Keyword	Description
as <user-name> identified by <password>	Provide the name and password of a default partition user who can connect to both databases. Essbase uses the login information to synchronize database outlines.
using <user-name> identified by <password> for creation	Create the partition using a different user than the one being set as the default partition user. This can be useful when you want to specify a read-only user account as the default partition user.
mapped...	Define the member-name mapping for shared sections of both databases, if member names for sections that map are different in the two databases.
outline...	Specify the direction in which outline synchronization should proceed, if necessary. The default direction is the same as the data-refresh direction.
default login as	Specify a default user name and password with which to provide generic access to the linked-partition data source. When accessing a linked partition, Essbase attempts to use the end user's login information to connect to the source database. If the user does not have access to the source database, Essbase looks for the linked-partition default user name and password.
comment	Create a comment to describe the source half of the partition definition.
remote comment	Create a comment to describe the target half of the partition definition.
validate only	Validate the existing partition definition described by this statement, without actually creating it.

Notes

- Multiple area specifications are allowed, provided they are separated by whitespace. Multiple mappings are allowed, provided they are separated by whitespace. All area aliases used in a mapping should be associated with the target, and the direction of the mapped clause should go from source to target.
- The first DBS-NAME is the local database, and the second DBS-NAME is the remote database.
- Creating a partition *to* the remote site means the current database is the source. Creating a partition *from* the remote site means the current database is the target.
- If you are creating a partition and specifying a host name that includes a port number, see [“Specifying Port Numbers in Partition Host Names” on page 923](#) for more information.
- If you are using host name aliases, see [“Using Host Name Aliases When Partitioning” on page 924](#).
- Aggregate storage databases can be the source, the target, or the source and target of a linked partition. Outline synchronization (**refresh outline** statement) is not currently enabled for partitions that involve aggregate storage databases.
- To create a partition as an externally authenticated user, when using MaxL Script Editor, you must enter a **login** statement before the **create partition** statement. The login statement must include the full external user name with provider, as well as the host name.

For example,

```
login 'admin@Native Directory' 'password' on 'FQN';
create partition....;
```

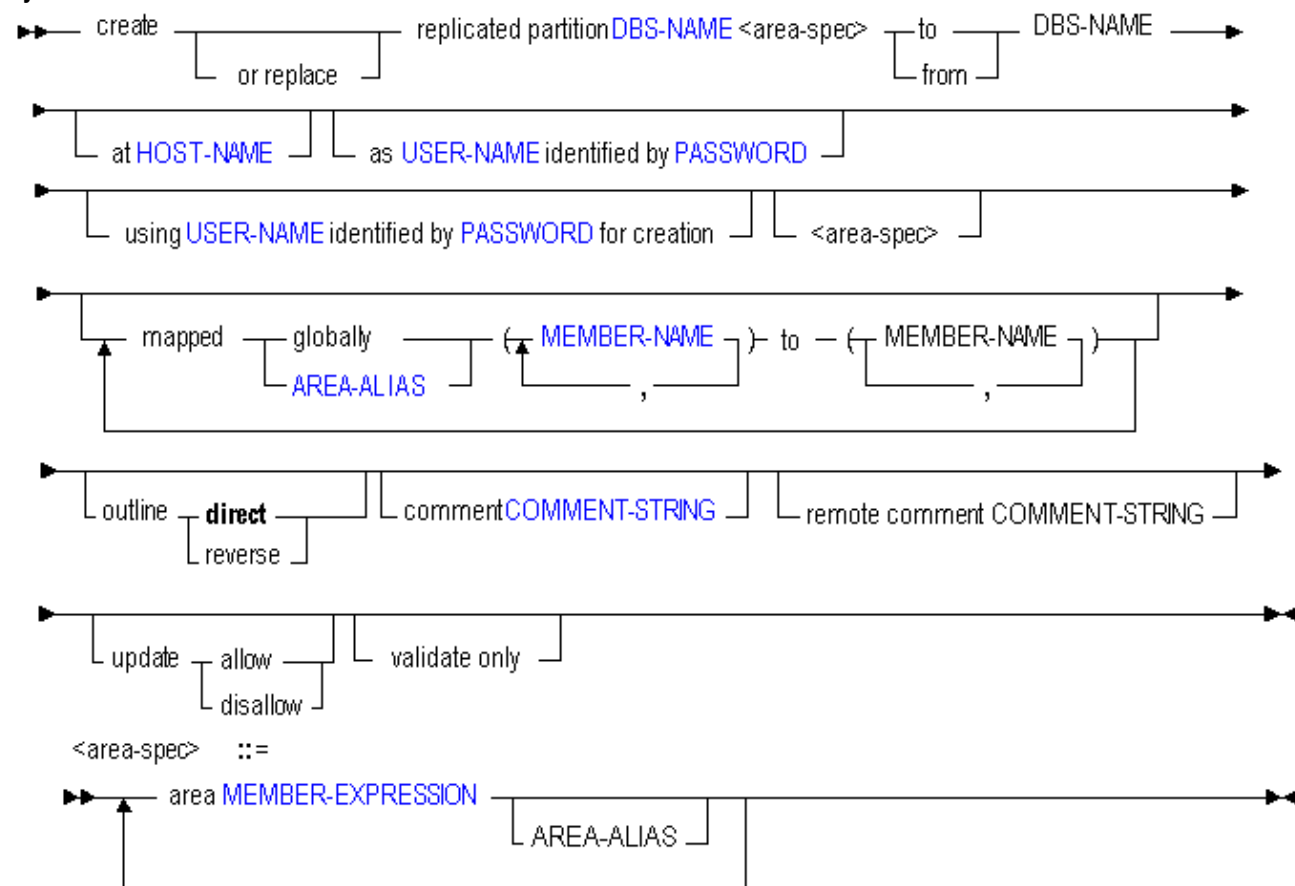
Example

```
create or replace linked partition sampeast.east
  area '@DESCENDANTS("Eastern Region"), @DESCENDANTS(Qtr1) '
to samppart.company at localhost
as partitionuser identified by 'password'
  area '@DESCENDANTS(East) @DESCENDANTS(Qtr1) '
  area '"Region 9020" "FLD Other"'
default login as appdesigner identified by 'password';
```

Create Replicated Partition

Create or validate a replicated partition definition between two databases. A replicated partition copies a portion of the source (or master) database to be stored in a target database. Users can access the target database as if it were the source. The administrator must periodically [refresh](#) the target data from the source data.

Syntax



Use **create replicated partition** to create a partition in the following ways:

Keyword	Description
create replicated partition	Create a replicated partition. A replicated partition is a copy of a portion of the data source that is stored in the data target.
create or replace ...partition	Create a partition definition, or replace an existing partition definition.
area...	Define the partition areas to share with the other database. Optionally nickname the area using an area-alias .
to <db-name>	Create a partition definition between the current database source and the second database (the target).
from <db-name>	Create a partition definition between the current database target and the second database (the source).
at <host-name>	Specify the remote computer name, if you are creating a partition definition between the current database and one residing on a remote Essbase Server host.
as <user-name> identified by <password>	Provide the name and password of a default partition user who can connect to both databases. Essbase uses the login information to: <ul style="list-style-type: none"> ● Transfer data between the source and the target for replicated and transparent partitions. Database security filters can be applied to prevent end users from seeing privileged data. ● Synchronize database outlines for all partition types.
using <user-name> identified by <password> for creation	Create the partition using a different user than the one being set as the default partition user. This can be useful when you want to specify a read-only user account as the default partition user.
mapped...	Define the member-name mapping for shared sections of both databases, if member names for sections that map are different in the two databases.
outline...	Specify the direction in which outline synchronization should proceed, if necessary. The default direction is the same as the data-refresh direction.
update...	Allow or disallow the updating of data in a replicated-type partition target. If you do not specify update allow, by default, the replicated partition cannot be updated.
comment	Create a comment to describe the source half of the partition definition.
remote comment	Create a comment to describe the target half of the partition definition.
validate only	Validate the existing partition definition described by this statement, without actually creating it.

Notes

- Multiple area specifications are allowed, provided they are separated by whitespace. Multiple mappings are allowed, provided they are separated by whitespace. All area aliases used in a mapping should be associated with the target, and the direction of the mapped clause should go from source to target.
- The first DBS-NAME is the local database, and the second DBS-NAME is the remote database.

- Creating a partition *to* the remote site means the current database is the source. Creating a partition *from* the remote site means the current database is the target.
- If you are creating a partition and specifying a host name that includes a port number, see [“Specifying Port Numbers in Partition Host Names” on page 923](#) for more information.
- If you are using host name aliases, see [“Using Host Name Aliases When Partitioning” on page 924](#).
- Aggregate storage databases can be the target, but not the source, of a replicated partition.
- To create a partition as an externally authenticated user, when using MaxL Script Editor, you must enter a **login** statement before the **create partition** statement. The login statement must include the full external user name with provider, as well as the host name.

For example,

```
login 'admin@Native Directory' 'password' on 'FQN';
create partition....;
```

Example

```
create or replace replicated partition source.source
area 'DimensionA' sourceAreaA
area 'DimensionB' sourceAreaB
to target.target at localhost
as admin identified by 'password'
area 'ParentMemberA' targetAreaA
area 'ParentMemberB' targetAreaB
mapped targetAreaA (ChildA) to (Child_a)
mapped targetAreaB (ChildB) to (Child_b)
;
```

Creates a partition from database Source to database Target where the partitioned areas between them are DimensionA and DimensionB on the source, corresponding to ParentMemberA and ParentMemberB (respectively) on the target. Differences in member names between the two partitioned areas are resolved during the partition creation, using the *mapped* clauses. Area aliases are used after each area specification, so that members can be mapped specifically for each area.

```
create or replace replicated partition sampeast.east
area '@IDESCENDANTS("Eastern Region"), @IDESCENDANTS(Qtr1)'
to samppart.company at localhost
as partitionuser identified by 'password'
area '@IDESCENDANTS(East) @IDESCENDANTS(Qtr1)'
update disallow;
```

Creates a replicated partition from an area in the source database, sampeast.east, to an area in the target database, samppart.company.

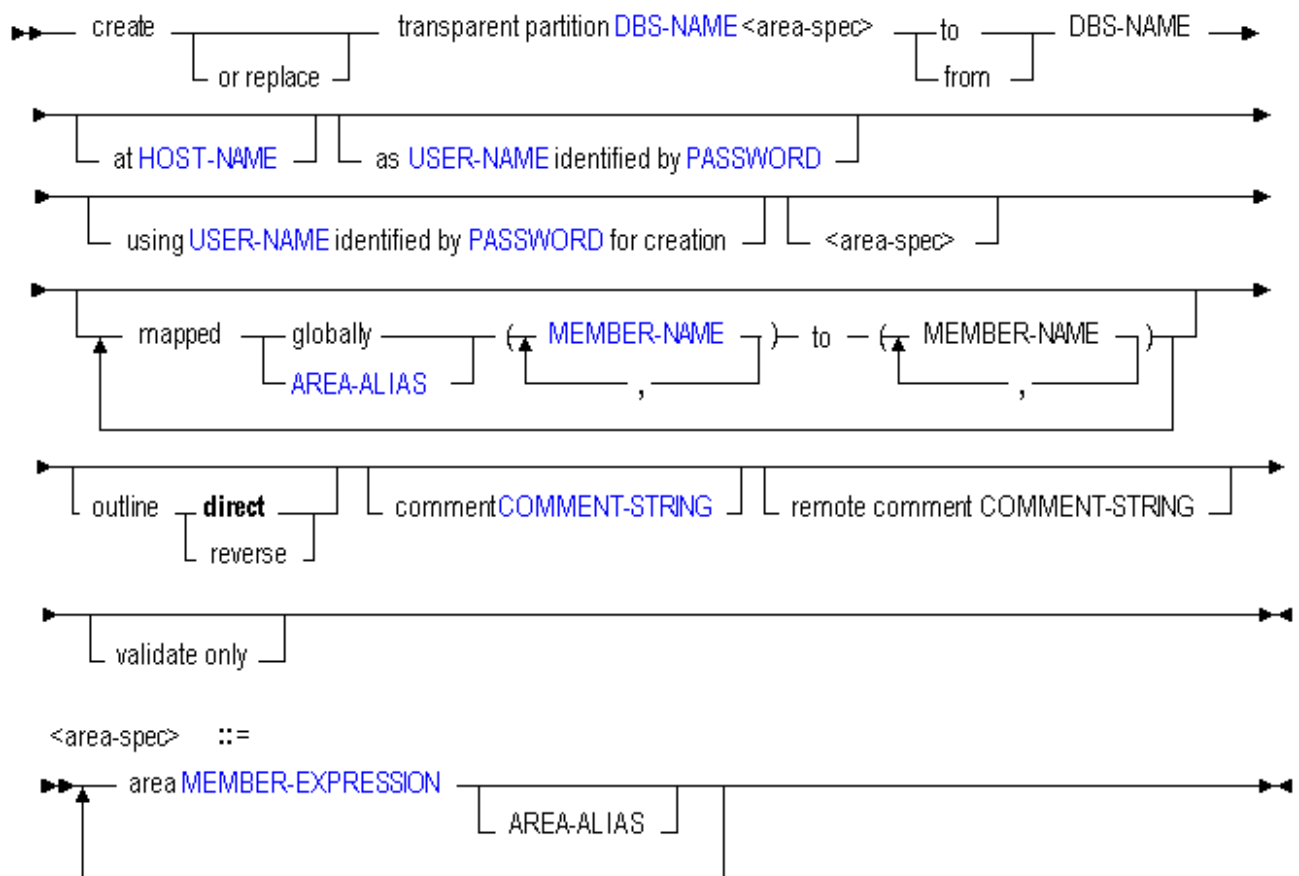
```
create or replace replicated partition sampeast.east
area '@IDESCENDANTS("Eastern Region"), @IDESCENDANTS(Qtr1)'
to samppart.company at localhost
as admin identified by 'password'
area '@IDESCENDANTS(East) @IDESCENDANTS(Qtr1)' foo
mapped foo (Year) to (Yr)
update allow validate only;
```


Validates the syntax of a replicated partition you might want to create. To create the partition after checking validity, simply remove the *validate only* phrase. For an explanation of *foo* as used above, see the definition for “[AREA-ALIAS](#)” on page 773 .

Create Transparent Partition

Create or validate a transparent partition definition between two databases. A transparent partition allows users to manipulate data that is stored in a target database as if it were part of the source database. The remote data is retrieved from the data source each time that users at the data target request it.

Syntax



Use **create transparent partition** to create a partition in the following ways:

Keyword	Description
create transparent partition	Create a transparent partition. A transparent partition enables users to access data from the data source as though it were stored in the data target. The data is, however, stored at the data source, which can be in another application, in another database, or on another Essbase Server.
create or replace ...partition	Create a partition definition, or replace an existing partition definition.
area...	Define the partition areas to share with the other database. Optionally nickname the area using an area-alias .

Keyword	Description
to <db-name>	Create a partition definition between the current database source and the second database (the target).
from <db-name>	Create a partition definition between the current database target and the second database (the source).
at <host-name>	Specify the remote computer name, if you are creating a partition definition between the current database and one residing on a remote Essbase Server host.
as <user-name> identified by <password>	Provide the name and password of a default partition user who can connect to both databases. Essbase uses the login information to: <ul style="list-style-type: none"> ● Transfer data between the source and the target for replicated and transparent partitions. Database security filters can be applied to prevent end users from seeing privileged data. ● Synchronize database outlines for all partition types.
using <user-name> identified by <password> for creation	Create the partition using a different user than the one being set as the default partition user. This can be useful when you want to specify a read-only user account as the default partition user.
mapped...	Define the member-name mapping for shared sections of both databases, if member names for sections that map are different in the two databases.
outline...	Specify the direction in which outline synchronization should proceed, if necessary. The default direction is the same as the data-refresh direction.
comment	Create a comment to describe the source half of the partition definition.
remote comment	Create a comment to describe the target half of the partition definition.
validate only	Validate the existing partition definition described by this statement, without actually creating it.

Notes

- Multiple area specifications are allowed, provided they are separated by whitespace. Multiple mappings are allowed, provided they are separated by whitespace. All area aliases used in a mapping should be associated with the target, and the direction of the mapped clause should go from source to target.
- The first DBS-NAME is the local database, and the second DBS-NAME is the remote database.
- Creating a partition *to* the remote site means the current database is the source. Creating a partition *from* the remote site means the current database is the target.
- If you are creating a partition and specifying a host name that includes a port number, see [“Specifying Port Numbers in Partition Host Names” on page 923](#) for more information.
- If you are using host name aliases, see [“Using Host Name Aliases When Partitioning” on page 924](#).
- Aggregate storage databases can be the source, the target, or the source and target of a transparent partition. Outline synchronization (**refresh outline** statement) is not currently enabled for partitions that involve aggregate storage databases.

- To create a partition as an externally authenticated user, when using MaxL Script Editor, you must enter a **login** statement before the **create partition** statement. The login statement must include the full external user name with provider, as well as the host name.

For example,

```
login 'admin@Native Directory' 'password' on 'FQN';
create partition....;
```

Example

```
create or replace transparent partition sampeast.east
    area '@CHILDREN("Eastern Region"), @CHILDREN(Qtr1)' sourceArea
to samppart.company at localhost
as partitionuser identified by 'password'
    area '@CHILDREN(East) @CHILDREN(Qtr1)' targetArea;
```

Creates a transparent partition between the source, sampeast.east, and the target, samppart.company. The partition is defined only for the areas specified by the area aliases sourceArea and targetArea.

```
create or replace transparent partition source.source
    area 'DimensionA' sourceAreaA
    area 'DimensionB' sourceAreaB
to target.target at localhost
as admin identified by 'password'
    area 'ParentMemberA' targetAreaA
    area 'ParentMemberB' targetAreaB
mapped targetAreaA (ChildA) to (Child_a)
mapped targetAreaB (ChildB) to (Child_b)
;
```

Creates a partition from database Source to database Target where the partitioned areas between them are DimensionA and DimensionB on the source, corresponding to ParentMemberA and ParentMemberB (respectively) on the target. Differences in member names between the two partitioned areas are resolved during the partition creation, using the *mapped* clauses. Area aliases are used after each area specification, so that members can be mapped specifically for each area.

Create Trigger

Create or replace a trigger to track state changes over a selected cube area.

Select the type of trigger to create:

- [on-update](#)
- [after-update](#)

Create After-Update Trigger

Create or replace a trigger to track state changes over a selected cube area.

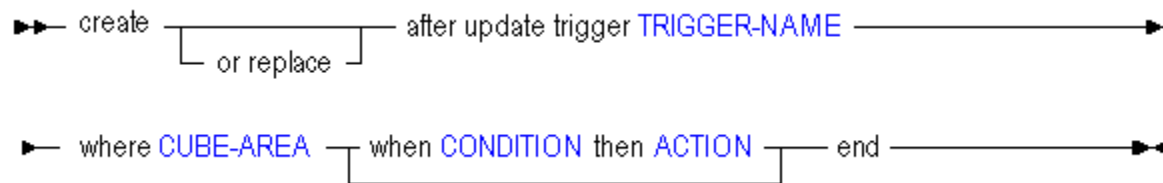
Triggers help you track whether designated constraints are violated during updates (events) in the area, and allow you to specify resultant actions to execute if violations are detected. Minimum permission required: Database Manager.

Create an *after-update* trigger if you want the trigger to be activated after the entire data update operation is completed. This is the only type of trigger supported in aggregate storage mode. When after-update triggers are used, the trigger fires when an update operation on level-0 data cells is complete, and the update operation as a whole has met any condition specified for the cube area.

For more information about the Essbase triggers feature, see the *Oracle Essbase Database Administrator's Guide*.

Note: You cannot create or replace a trigger during a calculation, or a data load (including a lock and send).

Syntax



Use **create after update trigger** to create a trigger in the following ways:

Keyword	Description
create after update trigger	Create a new after-update trigger.
create or replace after update trigger	Create an after-update trigger, or replace an existing trigger of the same name.
where <cube area>	Define the area of the database to be tracked. Use a valid, symmetric MDX slicer specification .
when <condition>	Define the condition to be tested for using the keyword WHEN followed by a valid MDX conditional expression.
then <action>	Define the action to be taken if the WHEN condition is met. See examples in “Examples of Triggers” on page 927 .
end	The END keyword must terminate every create trigger statement.

Example

```

create or replace after update trigger Sample.Basic.EastColas
where (Jan, Sales, Actual, [100], East)
when Jan > 20 then spool EastColas_Fail end;
  
```

Logs a message in the \$ARBORPATH\app\Sample\Basic\trig\EastColas_Fail file.

Create On-Update Trigger

Create or replace an on-update trigger to track state changes over a selected cube area.

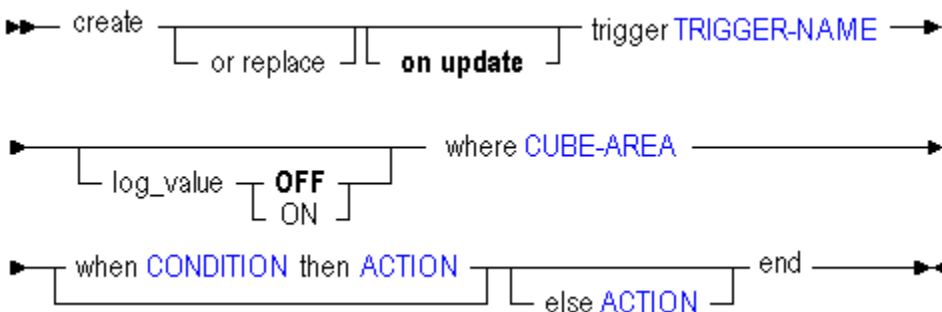
Triggers help you track whether designated constraints are violated during updates (events) in the area, and allow you to specify resultant actions to execute if violations are detected. Minimum permission required: Database Manager.

An *on-update* trigger is the default type of trigger, even if no type is specified. During a data update process, any cell update that meets a condition specified for the cube area will immediately activate the trigger. On-update triggers are not supported in aggregate storage databases. If you are using an aggregate storage database, you can create [after-update triggers](#).

For more information about the Essbase triggers feature, see the *Oracle Essbase Database Administrator's Guide*.

Note: You cannot create or replace a trigger during a calculation, or a data load (including a lock and send).

Syntax



Use **create on update trigger** to create a trigger in the following ways:

Keyword	Description
create [on update] trigger	Create a new on-update trigger. The on update keywords are optional; an on-update trigger is created by default.
create or replace [on update] trigger	Create an on-update trigger, or replace an existing trigger of the same name.
log_value OFF	Optional. Log no data values to the trigger spool file. This is the default.
log_value ON	Optional. Log new and old data values to the trigger spool file.
where <cube area>	Define the area of the database to be tracked. Use a valid, symmetric MDX slicer specification .
when <condition>	Define the condition to be tested for using the keyword WHEN followed by a valid MDX conditional expression.
then <action>	Define the action to be taken if the WHEN condition is met. See examples in “Examples of Triggers” on page 927 .

Keyword	Description
else <action>	Optional. Define an action to be taken if the WHEN condition is <i>not</i> met. See examples in “Examples of Triggers” on page 927.
end	The END keyword must terminate every create trigger statement.

Example

```
create or replace on update trigger Sample.Basic.EastColas
where (Jan, Sales, Actual, [100], East)
when Jan > 20 then spool EastColas_Fail end;
```

Logs a message in the \$ARBORPATH\app\Sample\Basic\trig\EastColas_Fail file.

Create User

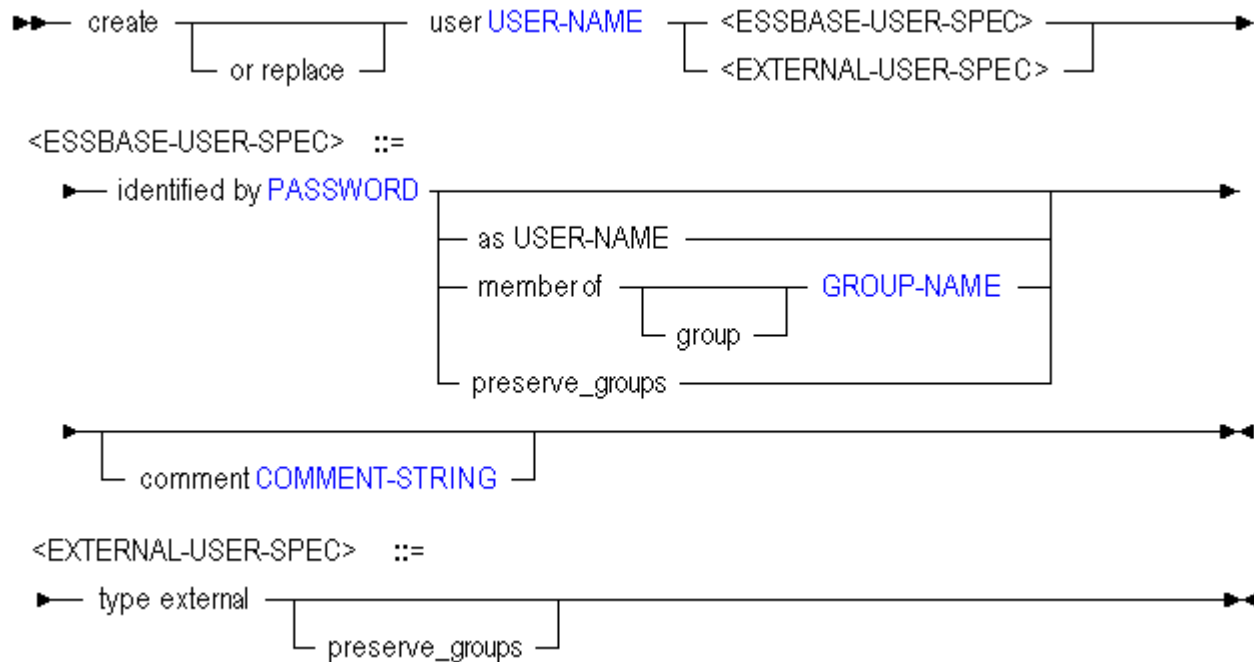
Create or re-create a user, either from scratch or as a copy of another user. Users can be created to log in using Essbase security. Optionally, create the Essbase-authenticated user as a member of a group.

Permission required: create_user, unless Essbase is in Shared Services mode.

In Shared Services mode, an Essbase user can be created in two ways:

1. From Essbase using MaxL or Administration Services. To do this, you must be an Essbase administrator. You must additionally have the Shared Services Directory Manager role.
2. From Shared Services. To do this, you must be a Shared Services administrator.

Syntax



Use create user to create a user in the following ways:

Keyword	Description
create user	Create a new Essbase user.
create or replace user	Create a new Essbase user. If a user of that name already exists, it is replaced.
create user as	Create a user as a copy of an existing user. The new user has an identical security profile to the user that was copied.
member of group	Create a user and assign membership to a security group.
preserve_groups	When replacing a user, preserve the original user's group associations.
comment	Create an optional comment to describe the user.
type external	Create a user that must log in using the Shared Services security platform. In order for the user to be able to log in successfully, the “AUTHENTICATIONMODULE” on page 395 parameter must be set to CSS in the <code>essbase.cfg</code> file, and the name must match a valid user name in the external authentication repository.

Example

```
create user Fiona identified by sunflower;
```

Creates a user called Fiona with the password sunflower.

```
create user Guest identified by 'password' member of group Visitors;
```

Creates a user called Guest with the password password, and adds Guest to the group called Visitors. Quotation marks are required because **password** is a MaxL keyword.

```
create or replace user Guest identified by 'password' as RecycleMe;
```

Creates a user called Guest as a copy of an existing user called RecycleMe. If Guest already exists, it is overwritten.

```
create or replace user 'Autumn Smith' type external;
```

Creates a user called Autumn Smith who is externally authenticated in a corporate authentication repository supported by the Shared Services security platform.

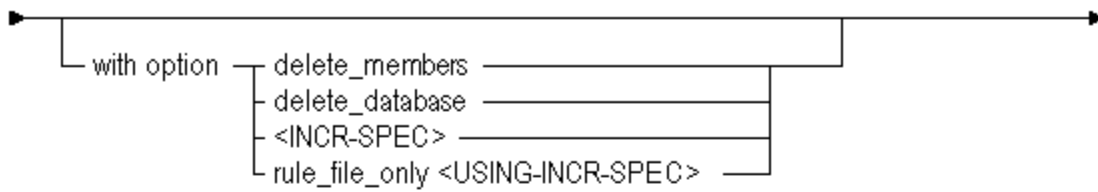
Deploy

Deploy a cube to the Essbase Server.

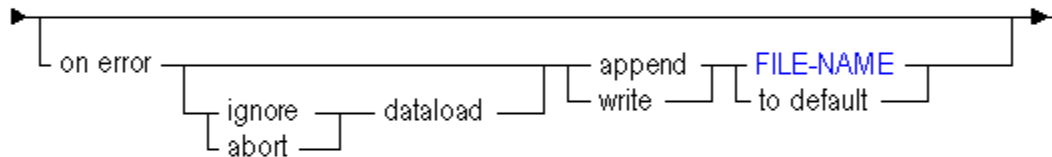
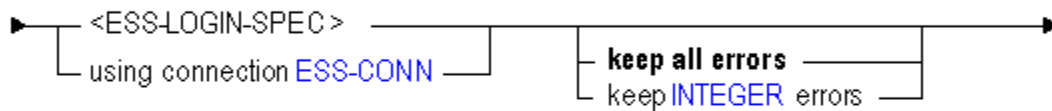
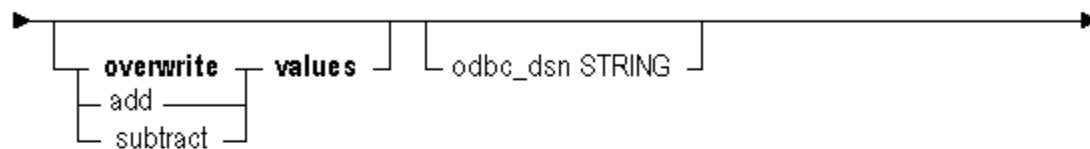
This MaxL Shell statement replicates the behavior of the Essbase Studio Cube Deployment Wizard.

For detailed information about cube deployment using Essbase Studio, see the *Oracle Essbase Studio User's Guide*.

Syntax



▶ <STUDIO-LOGIN-SPEC> — to application **APP-NAME** on database **DBS-NAME** →



<STUDIO-LOGIN-SPEC> ::=



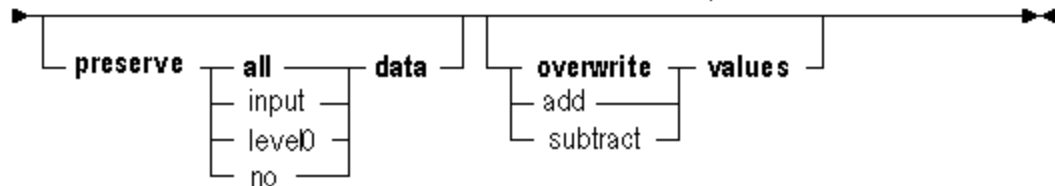
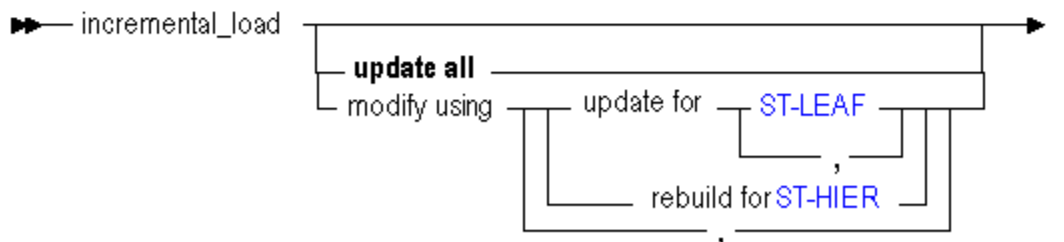
<ESS-LOGIN-SPEC> ::=



<USING-INCR-SPEC> ::=



<INCR-SPEC> ::=



Use **deploy** to deploy a cube in the following ways:

Keyword	Description
Deploy [all] from model ...	Load dimensions and members to an Essbase outline, and then populate the Essbase database with data.
Deploy outline from model ...	Load only dimensions and members (without data) to an Essbase outline.
Deploy data from model ...	Populate Essbase database outline with data. Loading data establishes actual values for the cells defined by the structural outline of the database.
...in cube schema	Deploy the model from a cube schema in Essbase Studio.
...delete_members...	Remove all dimensions and members in an existing Essbase outline. When you delete members, Essbase Studio removes all members from the existing Essbase database outline and then uses the dimensions and members in the Essbase model to recreate the outline. Deploy operations can take longer when the delete_members keywords are used. Oracle recommends using this option only if you have a specific reason to do so. Use the delete_members keywords if, for example, you know that some members have been removed from the underlying hierarchies used to create an existing Essbase model.
...delete database...	Delete all members and data in the Essbase database before performing a member load, or a member and data load. This action clears the Essbase database outline before the outline build occurs, significantly reducing the amount of time required for the load. Do not use the delete database keywords if you are using the deploy data from model keywords.
...incremental_load ...	Update specific dimensions or members in the Essbase outline. You can perform incremental loads when loading members only, or when loading members and data. This keyword does not apply when only loading data.
...rule_file_only ...	Specify the changes Essbase should make to data and members from a data source while loading them into the Essbase database. The data source is not changed. Rules files are saved to the <code>app</code> directory of your Essbase installation.
to application... database	The application and database name of the cube to deploy.
...incremental_load [update all]	Update all hierarchies in the model. Any new members are added. When this phrase is used, all hierarchies are automatically selected for update.
...incremental_load modify using update for ...	Update specified hierarchies. Any new members are added to the specified hierarchies; existing members are retained and updated. Use this option to add new members without changing the hierarchy's structure, or to add shared members. During incremental update, an existing hierarchy is updated without removing the existing members.
...incremental_load modify using rebuild for ...	Rebuild specified hierarchies. Clears all the members of the specified hierarchies and adds back all members, including shared members. If necessary, restructures the hierarchy. This phrase is particularly useful if you have removed members from a hierarchy. Then the members that still exist, plus any new ones, are added back into the hierarchy and, if necessary, the hierarchy is restructured.

Keyword	Description
... incremental_load... preserve...all	Restructure the database during member load and preserve all existing data that applies to the changed outline when restructuring occurs.
... incremental_load... preserve...input	Restructure the database during member build and preserve only those blocks containing data that is loaded. Many applications contain data that is entered at parent levels. Using the preserve input keywords prevents deletion of any blocks that are created by data load, whether they are non-level zero or level zero (leaf node) blocks.
... incremental_load... preserve...level0	Restructure the database during member build and preserve data only for level zero members. This is the optimal restructure option if you change the source database and need to recalculate the data, and if all data required for the calculation is in level zero members. Using this keyword deletes all upper-level blocks before restructuring. This reduces the disk space required for restructuring and improves calculation time when the database is recalculated. The upper-level blocks are recreated when you calculate the database.
... incremental_load... preserve...no	Clear all data from the database.
... incremental_load... add	Add the values in the data source to the existing values in the cube.
... incremental_load... subtract	Subtract the values in the data source from the existing values in the cube.
... incremental_load... overwrite values	Replace the values in the cube with the values in the data source.
odbc_dsn	Provide a ODBC DSN name. If you choose to deploy using an ODBN DSN name in order to take advantage of your own custom ODBC DSN parameter settings, follow these guidelines: <ul style="list-style-type: none"> ● Set up your ODBC DSN before beginning deployment, on the server machine where Essbase is installed. ● The ODBC DSN must have the same user name and password as the data source connection being used in this deployment. <p>To use an Oracle OCI connect identifier, use the following syntax after odbc_dsn keyword:</p> <pre>\$OCI\$host:port/SID</pre> <p>Following is an example OCI connect identifier where the host server name is “myserver,” the port number is 1521, and the Oracle SID (Service Identifier) is “orcl”:</p> <pre>\$OCI\$myserver:1521/orcl</pre>
STUDIO-LOGIN-SPEC	Provide the name and password of the Essbase Studio user.
ESS-LOGIN-SPEC	Provide the name and password of an Essbase user who can create databases, and the name of the Essbase Server machine to which you want to deploy.
...using connection ...	Provide the name of a valid Essbase connection created in Essbase Studio.

Keyword	Description
...keep all INTEGER errors...	Keep all rejected records in the error file, or keep a specified number rejected records.
...on error ignore abort dataload ...	Choose either to ignore any errors during the data load process, or cancel the data load if there is an error.
...on error append write	If there is a deployment error, either add errors to an existing error file, or create a new one.
...FILE-NAME to default...	Specify an error file-name and path, or accept the default error file location at HYPERION_HOME/EssbaseStudio/server/essjapihome/data and the default file name of the following format: <i>app_name.db_name_timestamp.err</i>

Example

```
deploy all from model 'cs1Model' in cube schema '\CubeSchemas\cs1' login $1 identified
by $2 on host 'poplar-pc1' to application 'cs2' database 'cs2' add values using
connection 'Connection1' keep 200 errors on error ignore dataload write to default;
```

```
deploy outline from model 'MaxLModel3' with option incremental_load modify using rebuild
for 'Time' preserve all data login 'admin' identified by 'password' on host 'localhost'
to application 'mxldemo2' database 'maxldemo' using connection 'Connection1';
```

Display Application

View information about current application-wide settings.

Syntax



Use **display application** to display application information in the following ways:

Keyword	Description
all	Display all applications on the system.
<app-name>	Display the named application.

Output Columns

Column	Description
application	String. Name of the application.
comment	String. Optional description of the application.

Column	Description
startup	TRUE or FALSE. Whether all users who have at least read permission can start the application.
autostartup	TRUE or FALSE. Whether the application starts when Essbase Server starts.
minimum permission	String. Minimum level of permission all users can have to databases in the application.
connects	TRUE or FALSE. Whether any user with a permission lower than Application Manager can make connections to the databases in this application which would require the databases to be started.
commands	TRUE or FALSE. Whether users with sufficient permissions can make read requests (or higher) to databases in the application.
updates	TRUE or FALSE. Whether users with sufficient permissions can make write requests (or higher) to databases in the application.
security	TRUE or FALSE. If FALSE, the Essbase security settings are disabled for the application, and all users are treated as Application Managers.
lock_timeout	Number. Maximum time interval (in seconds) that locks on data blocks can be held by clients.
max_lro_file_size	Number. If 0, there is no limit on the size of LRO attachments. All other sizes are displayed in kilobytes.
application_type	The type of encoding for the application. 0 Unspecified encoding type. The application was created using a pre-Release 7.0 version of Essbase. 1 This value is not in use. 2 Non-Unicode-mode application 3 Unicode-mode application
application_locale	The language of the character set in use by the application.
server	The name of the computer hosting the Essbase Server.
application_status	0 Not Loaded 1 Loading 2 Loaded 3 Unloading
elapsed_time	How long the application has been loaded.
users_connected	The number of users currently connected to the application.
storage_type	0 Default data storage 1 Multidimensional data storage 2 DB2 relational data storage 3 Oracle relational data storage 4 Aggregate storage 5.....Hybrid analysis
number_of_databases	How many databases are in the application namespace.

Example

```
display application;
```

Displays information about all applications on the system.

```
display application Sample;
```

Displays information about the Sample application.

Display Calculation

View a list of stored calculations on the system.

Syntax

```
▶▶ display calculation [all | CALC-NAME | on application APP-NAME | on database DBS-NAME] ▶▶
```

Use **display calculation** to display calculations in the following ways:

Keyword	Description
all	Display all stored calculations on the system.
<calc-name>	Display the named calculation.
on application	Display all calculations on the specified application.
on database	Display all calculations on the specified database.

Example

```
display calculation;
```

Display Database

View information about current database-wide settings.

Syntax

```
▶▶ display database [all | DBS-NAME | on application APP-NAME] [request_history] ▶▶
```

Use **display database** to display database information in the following ways:

Keyword	Description
all	Display information for all databases on the system.
<dbs-name>	Display information about the specified database.

Keyword	Description
on application	Display information about all databases on the specified application.
request_history	Display information about recent requests for the database. Information about the last three requests is returned.

Example

```
display database;
```

Displays information about all databases on the system.

```
display database Sample.Basic;
```

Displays information about the Sample.Basic database.

Display Disk Volume

View a list of currently defined disk volume definitions.

Syntax

```
▶▶ display disk volume [all | UNIQUE-VOL-NAME] [on database DBS-NAME] ▶▶
```

Use **display disk volume** to display disk volume information in the following ways:

Keyword	Description
all	Display all disk-volume definitions on the system.
<unique-vol-name>	Display a disk-volume definition by name.
on database	Display all disk-volume definitions associated with the specified database.

Notes

To manage disk volumes, use [alter database](#) (containing add, drop, and set disk volume).

Output Columns

The values returned for the *file type* field are numeric, and translate as follows:

Column	Description
1	Index
2	Data
3	Index and Data

Example

```
display disk volume;
```

Displays all (if any) disk volumes defined on the system.

```
display disk volume sample.basic.'vol3/hyperion/Essbase';
```

or

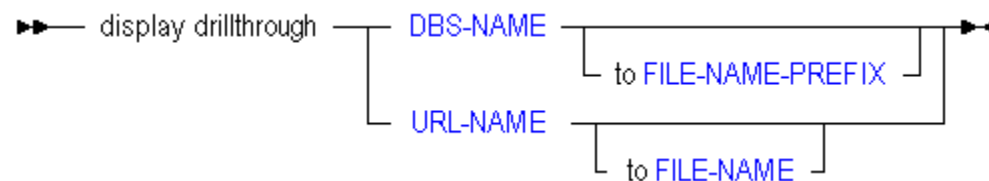
```
display disk volume sample.basic.C;
```

Displays information about a particular disk volume definition on Sample.Basic.

Display Drillthrough

View drill-through URL definitions used to link to content hosted on Oracle ERP and EPM applications.

Syntax



Use **display drillthrough** to display URL information in the following ways:

Keyword	Description
<dbs-name>	Display all drill-through URL definitions on the database. The number of drill-through URLs per database is limited to 255.
<dbs-name> to <file-name-prefix>	Display all drill-through URL definitions on the database, writing the URL XML content to file names prefixed with the string given as input for FILE-NAME-PREFIX.
<url-name>	Display the specified drill-through URL definition. The number of drillable regions in a drill-through URL is limited to 256. The number of characters per drillable region is limited to 65536.
<url-name> to <file-name>	Display the specified drill-through URL definition, writing the URL XML content to the specified file name.

Example

```
display drillthrough sample.basic;
```

Displays all drill-through URL definitions on Sample.Basic.

```
display drillthrough sample.basic to "urlxmls";
```

Displays all drill-through URL definitions on Sample.Basic, writing the URL XML content to file names prefixed with urlxmls.

```
display drillthrough sample.basic."Drill through To EPMI";
```

Displays the drill-through URL definition named `Drill through To EPMI`.

```
display drillthrough sample.basic."Drill through To EPMI" to "c:/temp/drillthrough.xml";
```

Displays the drill-through URL definition named `Drill through To EPMI`, writing the URL XML content to the file `drillthrough.xml`.

See Also

- [alter drillthrough](#)
- [create drillthrough](#)
- [drop drillthrough](#)

Display Filter

View a specific filter or a list of all filters on the system.

Syntax



Use **display filter** to display filters in the following ways. Use [display filter row](#) to display the contents of filters.

Keyword	Description
<code>all</code>	Display all filters on the system.
<code><filter-name></code>	Display a filter by name.
<code>on database</code>	Display all filters associated with the specified database.

Example

```
display filter;
```

Displays the names of all filters on the system.

Display Filter Row

View the filter rows which define database access within a specific filter or all filters.

Syntax



You can display filter contents in the following ways using **display filter row**.

Keyword	Description
all	Display all filters (and their contents) defined on the system.
<filter-name>	Display a filter and its contents by name.
on database	Display all filters (and their contents) associated with the specified database.

Example

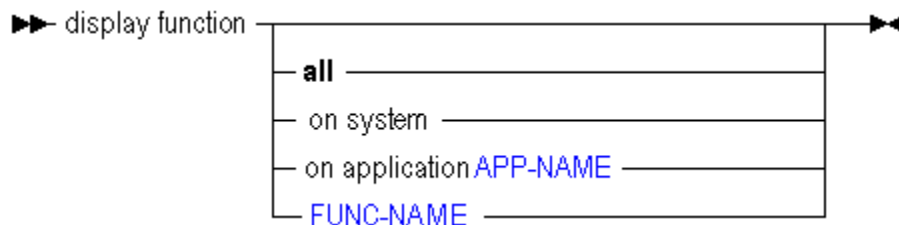
```
display filter row sample.basic.filt2;
```

Displays the row-by-row definition of a filter named `filt2` which is associated with `Sample.Basic`.

Display Function

View a list of custom-defined functions available globally or to an application. If MaxL shows no application name next to a function in the display output, then that function is global (system-wide). This statement also returns the validation status of an application's local custom-defined function or functions. Minimum permission required: read.

Syntax



Use **display function** to display custom-defined functions in the following ways:

Keyword	Description
all	Display all custom-defined functions, including those registered on the application level (local) or on the system level (global).
on system	Display all custom-defined functions registered on the system (global). Does not include locally defined functions.
on application	Display all custom-defined functions registered with the specified application (local). Does not include globally defined functions.

Keyword	Description
---------	-------------

<func-name> Display a custom-defined function by name.

Output Columns

The columns returned for this statement are described as follows:

Column	Description
--------	-------------

application Application name(s).

function Registered custom-defined function name(s), as defined by “[FUNC-NAME](#)” on page 786 in the [create function](#) statement.

class The java class before the method, as defined by “[JAVACLASS.METHOD](#)” on page 791 in the create function statement.

method The java method (at the end of the class), as defined by “[JAVACLASS.METHOD](#)” on page 791 in the create function statement.

spec Optional Essbase calculator-syntax specification string, as defined by “[CALC-SPEC-STRING](#)” on page 776 in the create function statement.

comment String as defined by “[COMMENT-STRING](#)” on page 777 in the create function statement.

runtime Values: TRUE or FALSE. Whether or not the custom-defined function was created with the **runtime** property.

state The current state of the registered custom-defined function.

Values:

- 0 = UNKNOWN. It is unknown whether the function is valid Java and is loaded into any application process.
- 1 = NOT_LOADED. The function is not loaded into any application process. You may have to [refresh](#) or restart the application in order to use this function. Or, the function may not be developed validly in Java.
- 2 = LOADED.
The function is valid Java, and is loaded into at least one application process.
- 3 = OVERRIDDEN. The local (application) function is overridden by a global (system-wide) function of the same name.

Example

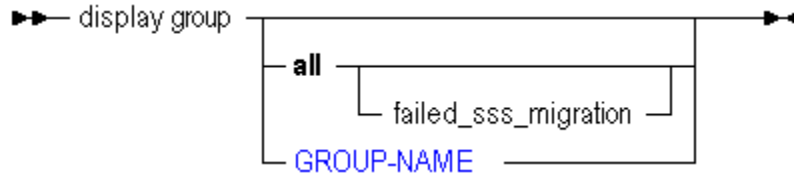
```
display function on application sample;
```

Displays all custom-defined functions associated with the application Sample.

Display Group

View a specific group or a list of all groups on the system. To view group membership information, use [display user](#).

Syntax



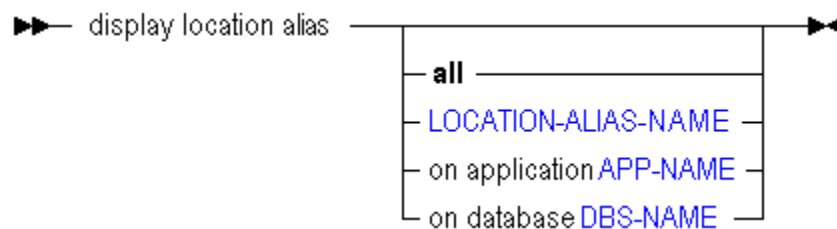
Use **display group** to display groups in the following ways:

Keyword	Description
all	Display all security groups on the system.
all failed_sss_migration	Display groups that did not successfully migrate to Shared Services when <code>alter system set sss_mode</code> or <code>alter group GROUP-NAME set sss_mode</code> was issued.
<group-name>	Display a security group by name.

Display Location Alias

View a specific location alias or a list of all location aliases defined on the system.

Syntax



You can display location aliases in the following ways using **display location alias**.

Keyword	Description
all	Display all location aliases defined on the system.
<location-alias-name>	Display a location alias by name.
on application	Display all location aliases defined for the specified application.
on database	Display all location aliases defined for the specified database.

Example

```
display location alias all;
```

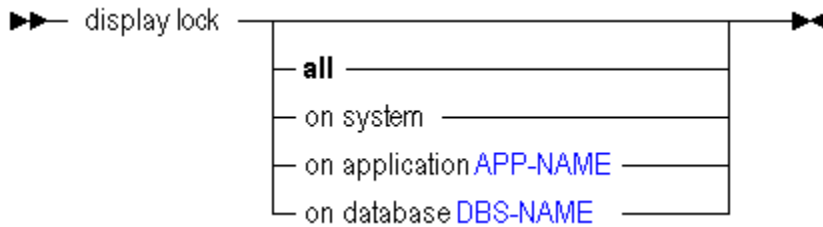
Displays a list of location aliases defined on the system.

Display Lock

View information about locks currently held by users or processes on data blocks.

Note: Data locks do not apply to aggregate storage applications.

Syntax



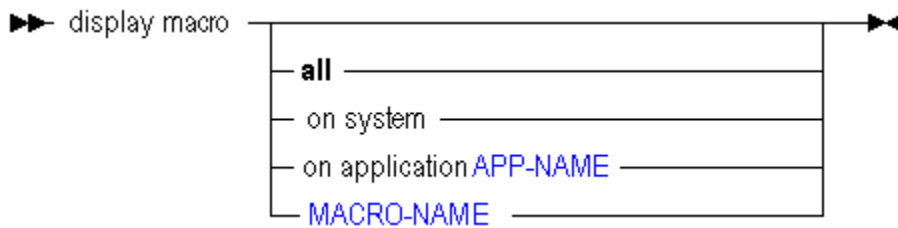
You can display locks in the following ways using **display lock**.

Keyword	Description
all	Display all locks on the specified scope. If all is omitted, this is the default.
on system	Display all locks on the system.
on application	Display all locks associated with the specified application.
on database	Display all locks associated with the specified database.

Display Macro

View a list of custom-defined macros available globally or to an application. Minimum permission required: read. If MaxL shows no application name next to a macro in the display output, then that macro is global (system-wide).

Syntax



You can display custom-defined macros in the following ways using **display macro**.

Keyword	Description
all	Display all custom-defined macros, including those registered on the application level (local) or on the system level (global).
on system	Display all custom-defined macros registered on the system (global). Does not include locally defined macros.
on application	Display all custom-defined macros registered with the specified application (local). Does not include globally defined macros.
<macro-name>	Display a custom-defined macro by name.

Output Columns

The columns returned for this statement are described as follows:

Column	Description
application	Application name(s).
macro	Macro name(s), as defined by “ MACRO-NAME ” on page 793 in the create macro statement.
signature	Macro signature, as defined by the custom-defined macro input parameters in the create macro statement.
expansion	Macro expansion, as defined by “ MACRO-EXPANSION ” on page 792 in the create macro statement.
spec	Optional Essbase calculator-syntax specification string, as defined by “ CALC-SPEC-STRING ” on page 776 in the create macro statement.
comment	String as defined by “ COMMENT-STRING ” on page 777 in the create macro statement.
state	The current state of the registered custom-defined macro. Values: <ul style="list-style-type: none">● 0 = UNKNOWN. It is unknown whether the macro is loaded into any application process.● 1 = NOT_LOADED. The macro is not loaded into any application process. You may have to refresh or restart the application in order to use this macro.● 2 = LOADED. The macro is loaded into at least one application process.● 3 = OVERRIDDEN. The local (application) macro is overridden by a global (system-wide) macro of the same name.

Example

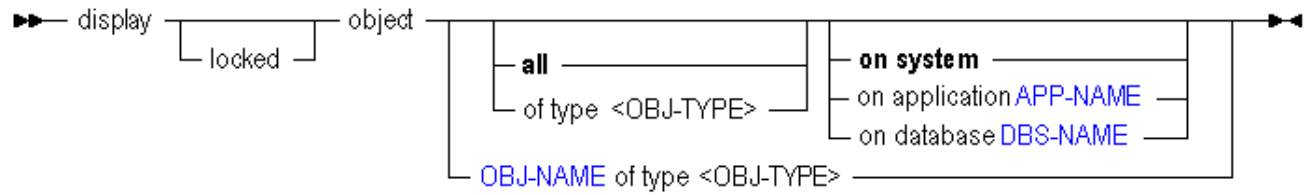
```
display macro on application sample;
```

Displays all custom-defined macros associated with the application Sample.

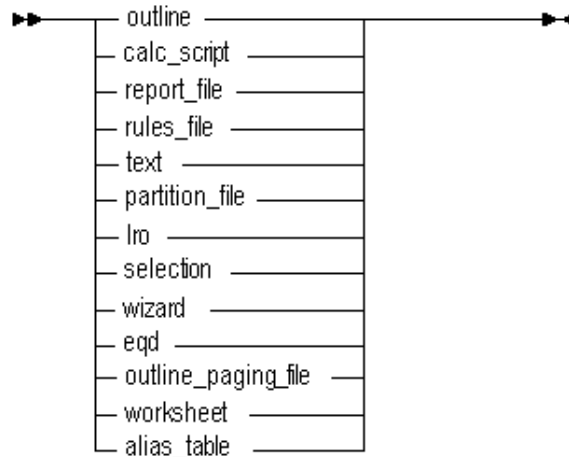
Display Object

View a list of database-related file objects stored in database directories.

Syntax



<OBJ-TYPE> ::=



You can display objects in the following ways using **display object**.

Keyword	Description
all	Display all stored objects on the specified scope.
locked	Display only locked objects on the specified scope.
of type...	Display only the objects of type specified by OBJ-TYPE ::=.
OBJ-NAME of OBJ-TYPE	Display a specific object by name and type.
on system	Display all stored objects on the system.
on application	Display all objects associated with the specified application.
on database	Display all objects associated with the specified database.

Example

```
MAXL> display object sample.basic.Calcdat of type text;
```

applicati	database	object_na	object_ty	locked	locked_by	locked_time
Sample	Basic	Calcdat	9	FALSE	N/A	N/A

Display Partition

View information about a specific partitioned database or all partitioned databases on the system. Only displays partition information for applications which are currently started.

Syntax



You can display partition information in the following ways using **display partition**.

Keyword Description

all	Display all partitions defined on the system.
on database	Display all partitions associated with the specified database.
advanced	Display full information including areas and member mappings for local and remote pieces of partitions.

Notes

If a partition definition is invalid, the same partition may be displayed twice, one time for each half. Each half will show the connection information of the other half.

Example

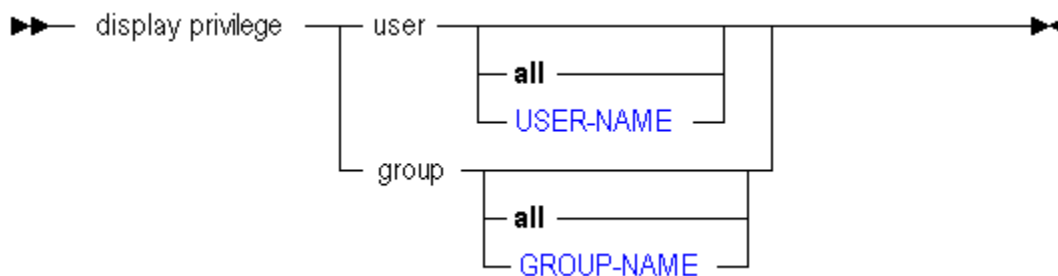
```
display partition all;
```

Displays information about all partitioned databases defined on the system.

Display Privilege

View a list of privileges, calculations, or filters held by users or groups.

Syntax



You can display security permissions in the following ways using **display privilege**.

Keyword Description

user...	Display security permissions for all users, or for a specified user.
group...	Display security permissions for all groups, or for a specified group.

Output Columns

The values returned for the *type* field are numeric, and translate as follows:

Column Description

- 1 “System-Level System Privileges” on page 814
- 2 “System-Level System Roles” on page 814
- 3 Execute calculation
- 4 Filter

Example

```
display privilege user Fiona;
```

Displays the privileges user Fiona has on each database object, including any calculations or filters granted to Fiona.

```
display privilege group;
```

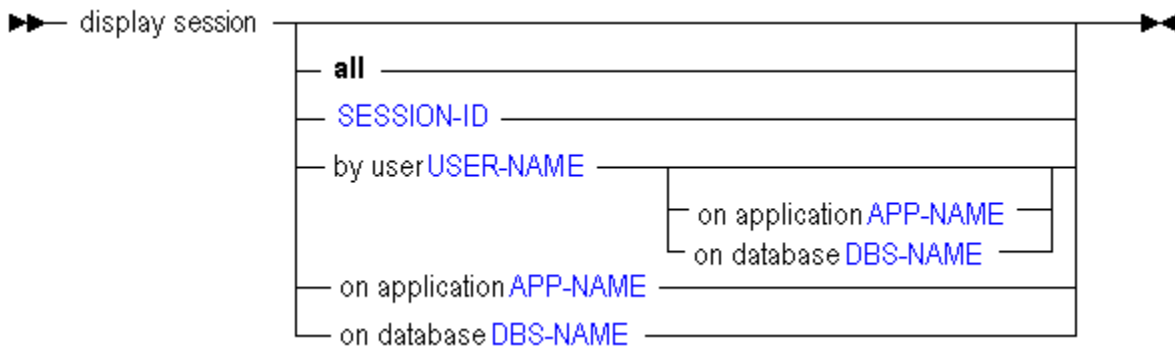
Displays privileges held by all groups on the system to all applications and databases on the system.

Display Session

View active login sessions on the current server, application, or database, including:

- The user that owns each session
- A session ID for each session
- How long the sessions have been active
- Information about outstanding requests (description, time started, name of computer originating the request, and status).

Syntax



You can display login and request information in the following ways using **display session**.

Keyword	Description
all	Display information about all current user sessions and active requests.
<session-id>	Display information about a particular user session, indicated by the numeric session ID.

Keyword	Description
by user	Display information about all current sessions by a particular user.
by user on application	Display information about all current sessions by a particular user on the specified application.
by user on database	Display information about all current sessions by a particular user on the specified database.
on application	Display information about all current sessions on the specified application.
on database	Display information about all current sessions on the specified database.

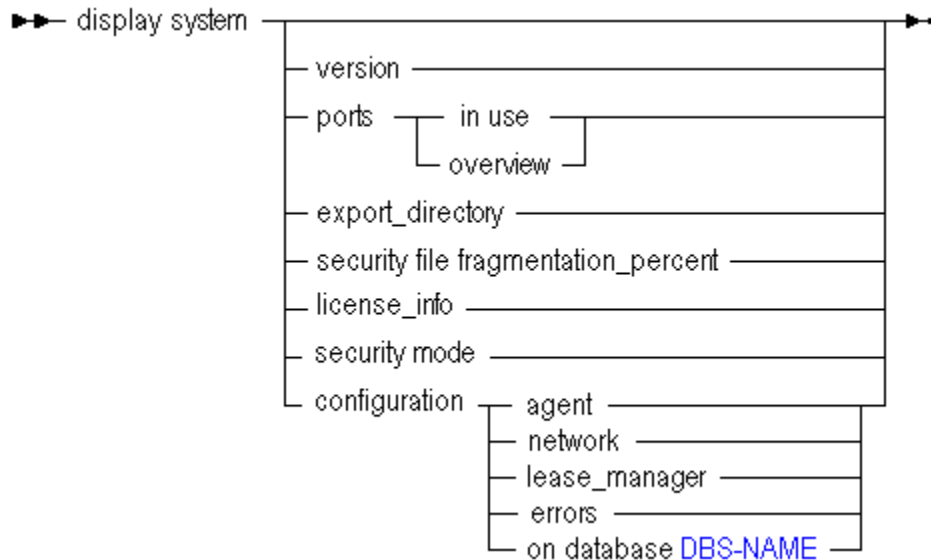
Example

```
display session;
display session on database sample.basic;
```

Display System

View information about current system-wide settings.

Syntax



You can display server-wide information in the following ways using **display system**.

Keyword	Description
display system	Display current connections and system-wide settings. configuration field values are numeric, and translate as follows: 2 Non-Unicode mode 3 Unicode mode
display system version	Display the server software version number.

Keyword	Description
display system ports in use	Display information about ports currently in use on the system.
display system ports overview	Display the number of ports that are available and in use on the system.
display system export_directory	<p>Display names of directories created for linked-reporting objects exported from a database to a directory created in \$ARBORPATH\app.</p> <p>If you used export lro and gave a full path to a directory for export files, those directories are not listed. Only export directories created in the ARBORPATH\App directory using the following export lro method are listed:</p> <pre>export database DBS-NAME lro to <server or local> directory DBS-EXPORT-DIR;</pre> <p>where DBS-EXPORT-DIR is a suffix (for example, dir1) for the name of a directory created by MaxL in \$ARBORPATH\App. MaxL creates the directory with a prefix of appname-dbsname-. For example, display system export_directory would list the following directories existing under \$ARBORPATH\App:</p> <pre>sample-basic-dir1 sample-basic-dir2</pre> <p>but it would not list export directories created elsewhere by providing a full directory path when using the export lro statement, such as: c:\MyExports\MyExportDir</p>
display system security file fragmentation_percent	<p>Display the percentage of security file fragmentation. 0% means the security file is not fragmented, and higher percentages indicate the degree of fragmentation.</p> <p>Fragmentation can gradually develop when objects such as users, groups, applications or databases are removed or changed. To prevent fragmentation, the security file is compacted each time the Agent shuts down.</p> <p>You can also defragment the security file without stopping the Agent. For more information, see</p> <ul style="list-style-type: none"> • The essbase.cfg setting “SECURITYFILECOMPACTIONPERCENT” on page 491. • The MaxL statement alter system compact security file; • The Agent command COMPACT (for documentation of Agent commands, see the <i>Oracle Essbase Database Administrator's Guide</i>).
display system license_info	Display information about the license settings implemented on the system.
display system security mode	<p>The type of security in use: native or Shared Services mode.</p> <p>security_mode field values are numeric, and translate as follows:</p> <pre>1 Native Essbase security 2 Shared Services security</pre>
display system configuration agent	Display values set using the <code>essbase.cfg</code> file, but display only values that apply to Essbase Agent. Permission required: administrator.
display system configuration network	Display values set using the <code>essbase.cfg</code> file, but display only values that apply to the network layer. Permission required: administrator.

Keyword	Description
display system configuration lease_manager	Display values set using the <code>essbase.cfg</code> file, but display only values that apply to lease manager. Permission required: administrator.
display system configuration errors	Display all lines in the <code>essbase.cfg</code> file that are errors: an error is any line entry that is not a comment <i>and</i> results in nothing being set. Permission required: administrator.
display system configuration on database DBS-NAME	Display values set using the <code>essbase.cfg</code> file, but display only values that apply to the named database. Permission required: administrator.

Example

```
display system;
```

Displays current password and session management settings.

```
display system configuration agent;
```

Displays current `essbase.cfg` settings that apply to the Essbase Agent.

Sample Outputs for Display System Configuration

```
MAXL> set column_width 40;
```

```
MAXL> display system configuration agent;
```

KEYWORDS	SETTINGS
AUTHENTICATIONMODULE	CSS
JVMMODULELOCATION	E:\Hyperion\common\JRE-64\Sun\1.5.0\bin
MAXLOGINS	50000
PORTUSAGELOGINTERVAL	600

OK/INFO - 1241044 - Records returned: [4].

```
MAXL> display system configuration network;
```

KEYWORDS	SETTINGS
AGENTPORT	1423
NETDELAY	1500
NETRETRYCOUNT	2000
SERVERPORTBEGIN	32768
SERVERPORTEND	33768

OK/INFO - 1241044 - Records returned: [5].

```
MAXL> display system configuration on database democfg.basic;
```

KEYWORDS	SETTINGS
CALCCACHE	TRUE
CALCCACHEDEFAULT	1250000
CALCCACHEHIGH	1750000

```

CALCCACHELOW                40000
CALCLOCKBLOCKDEFAULT        1000
CALCLOCKBLOCKHIGH           5000
CALCLOCKBLOCKLOW            500
CALCNOTICEDEFAULT           20
CALCNOTICEHIGH              50
CALCNOTICELOW               5
DATAERRORLIMIT              50000
DLSINGLETHREADPERSTAGE       FALSE
DLTHREADSPREPARE            4
DLTHREADSWRITE              4
DYNCALCCACHEMAXSIZE         DB[41943040], SV[41943040]
JVMMODULELOCATION             E:\Hyperion\common\JRE-64\Sun\1.5.0\bin
LOGMESSAGELEVEL              INFO
NOMSGLOGGINGONDATAERRORLIMIT TRUE
NUMERICPRECISION            1
SSLOGUNKNOWN                 FALSE
SSPROCROWLIMIT              250000

```

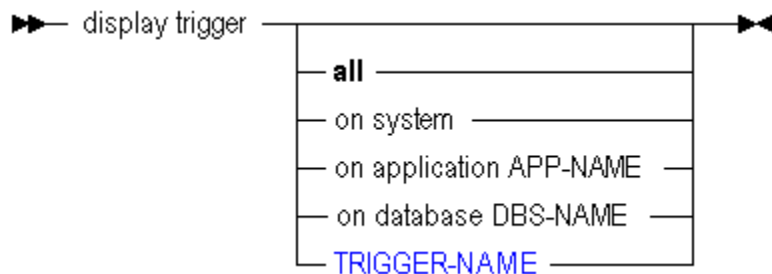
OK/INFO - 1241044 - Records returned: [21].

Display Trigger

View details about a trigger created to track state changes over a selected cube area. For more information about the Essbase triggers feature, see the *Oracle Essbase Database Administrator's Guide*.

Note: The application containing the trigger must be started in order to use display trigger.

Syntax



Output Columns

Column Description

application	The name of the application that contains the database.
database	The name of the database that contains the trigger. Essbase lists only databases that contain triggers.
name	The name of the trigger.
definition	The MaxL trigger statement (for example, create or replace trigger)
enabled	Whether Essbase is set to monitor the trigger. Values: TRUE or FALSE. To change the value, use alter trigger .

Example

display trigger on database Sample.Basic;

This example displays the output columns:

application	database	name	definition	enabled
Sample	Basic	WatchCosts	create or replace trigger	TRUE

Display Trigger Spool

View the log file created by a trigger. Triggers track state changes over a selected cube area. For more information about the Essbase triggers feature, see [Defining Triggers](#).

Syntax

```
▶▶ display trigger_spool [all | on application APP-NAME | on database DBS-NAME | SPOOL-NAME]
```

Display User

View a specific user or a list of all users defined on the system. View account and group membership information.

Syntax

```
▶▶ display user [all failed_sss_migration | USER-NAME | in group [all | GROUP-NAME]]
```

You can display user information in the following ways using `display user`.

Keyword	Description
all	Display information about all users on the system.

Keyword**Description**

all failed_sss_migration

Display users that did not successfully migrate to Shared Services when `alter system set sss_mode` or `alter group GROUP-NAME set sss_mode` was issued.

The following situations are common reasons for users to fail migration:

- The user account is disabled.
- The user name is the same as a group name in Shared Services.
- A user is externally authenticated but the authentication provider is not running.

If any users failed migration, you can retry the migration using `alter user all set sss_mode`.

For more information about user migration considerations, see the *Oracle Essbase Database Administrator's Guide* topic titled "Migrating Essbase to Shared Services."

Sample output for this statement:

```

user
+-----
ksmith
user1
user2

```

all shared_services_native with auto_password

Display the user names and passwords of Shared Services users that were migrated to Shared Services with the option to have their passwords generated automatically.

Sample output for this statement:

```

          user          password
+-----+-----+
server1      BgjK11fNo
server2      BgjK11fNo

```

Note: If the administrator designated a specific password for the migrated users, the password is not displayed.

Keyword**Description**

all
migr_modified_access

Display user database permissions that changed during migration to Shared Services.

In Shared Services, if an Essbase application contains multiple databases, the databases must have the same user security access levels. During migration to Shared Services, if a user has different access levels for two databases in the same application, the user is given the more restrictive access level for both databases.

The output columns for this statement are:

Output Column	Description
user	The user name.
application	Applications to which the user has access.
database	Databases to which the user has access.
pre_Shared_Services_migration_access	The user's access for a specific database before migration to Shared Services.
current_access	The user's access level after migration to Shared Services. Includes access acquired through groups and any other means.
filter	Filters assigned to the user.

The values returned for the `pre_Shared_Services_migration_access` and `current_access` fields are based on hexadecimal values but are displayed as decimal values, as follows:

```

0          No access
255       Full database access
272       Filter
273       Read
275       Write
279       Calc
280       Metaread
311       Database Manager
375       Create database
887       Application Manager
1911      Create application
4095     Full application and database access
65535    Administrator

```

Most roles are inclusive of other roles, but some additional combinations are possible. For example:

```

Read + Filter would be 273+272, or 545
Write + Filter would be 275 + 272, or 547

```

<user-name> Display information about the specified user.

in group all Display membership information for all groups on the system.

in group <group-name> Display membership information for the specified group.

Keyword	Description												
application_access_type	<p>Display the licensed application access type for a user.</p> <p>If a user is created in Planning, it automatically has an application access type of Planning; if a user is created in Essbase, it automatically has an application access type of Essbase.</p> <p>application_access_type field values are numeric, and translate as follows:</p> <table border="0"> <tr> <td>0</td> <td>No access</td> <td></td> </tr> <tr> <td>1</td> <td>Essbase access</td> <td></td> </tr> <tr> <td>2</td> <td>Planning access</td> <td></td> </tr> <tr> <td>3</td> <td>Essbase and Planning access</td> <td>(requires 2 licenses)</td> </tr> </table> <p>The application access type can be modified in Essbase using Alter User, or the Planning application access type can be modified through Oracle Hyperion Planning, Fusion Edition.</p>	0	No access		1	Essbase access		2	Planning access		3	Essbase and Planning access	(requires 2 licenses)
0	No access												
1	Essbase access												
2	Planning access												
3	Essbase and Planning access	(requires 2 licenses)											

Output Columns

Column	Description												
user	String. Name of the user.												
description	String. Optional description of the user.												
logged in	Values: TRUE or FALSE.												
password_reset_days	Integer. The number of days before the password expires, or 0 if no expiration is set.												
enabled	Values: TRUE if the user account is active, or FALSE if the account has been disabled by an administrator.												
change_password	Values: TRUE if the user must change the password at the next login; FALSE otherwise.												
type	Values: <table border="0"> <tr> <td>0</td> <td>User is set up using native Essbase security.</td> </tr> <tr> <td>1</td> <td>No longer used.</td> </tr> <tr> <td>3</td> <td>User is externally authenticated using Shared Services.</td> </tr> </table>	0	User is set up using native Essbase security.	1	No longer used.	3	User is externally authenticated using Shared Services.						
0	User is set up using native Essbase security.												
1	No longer used.												
3	User is externally authenticated using Shared Services.												
protocol	If the user is externally authenticated using Shared Services, this field contains the value CSS. This field is blank if the type field is 0 (the user is not externally authenticated).												
conn param	This field is blank.												
application_access_type	Values: <table border="0"> <tr> <td>0</td> <td>No access</td> <td></td> </tr> <tr> <td>1</td> <td>Essbase access</td> <td></td> </tr> <tr> <td>2</td> <td>Planning access</td> <td></td> </tr> <tr> <td>3</td> <td>Essbase and Planning access</td> <td>(requires 2 licenses)</td> </tr> </table> <p>See also Descriptions section.</p>	0	No access		1	Essbase access		2	Planning access		3	Essbase and Planning access	(requires 2 licenses)
0	No access												
1	Essbase access												
2	Planning access												
3	Essbase and Planning access	(requires 2 licenses)											

Example

```
display user;
```


Displays all users on the system and shows whether they are logged in, whether their accounts are enabled, and whether their passwords are set to expire.

```
display user in group;
```

Displays the membership information of all groups on the system.

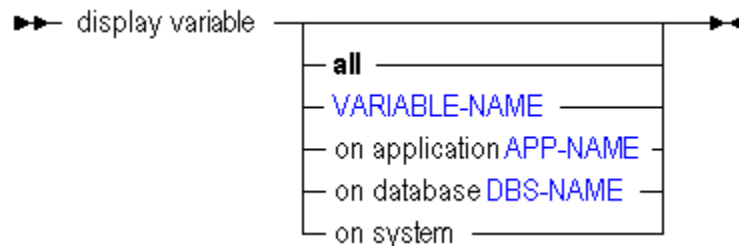
```
display user in group big_group;
```

Displays the membership information for a group called big_group.

Display Variable

View a list of substitution variables defined on the system.

Syntax



You can display substitution variables in the following ways using **display variable**.

Keyword	Description
all	Display all substitution variables defined on the Essbase Server, including those associated with applications and databases.
<variable-name>	Display a substitution variable by name. Permission required: read access for the applicable database or application. Administrator for system-defined variables.
on application	Display only substitution variables defined on the specified application. Permission required: read access for the application.
on database	Display only substitution variables defined on the specified database. Permission required: read access for the database.
on system	Display only the substitution variables associated with the Essbase Server. Permission required: Administrator.

Notes

To manage substitution variables, use [alter database](#) (containing add, drop, and set variable).

Example

```
display variable;
```

Displays a list of all substitution variables on the Essbase Server.

Drop Application

Delete an empty application from the system. To remove an application with databases, use **cascade**. To remove an application that has locked objects in a constituent database, you can use **force**. Minimum permission required: Application Manager.

Syntax

```
▶▶ drop application APP-NAME [ cascade ] [ force ] ▶▶
```

You can delete applications in the following ways using **drop application**.

Keyword Description

cascade Delete an application along with its constituent databases.

force Delete an application that may have locked objects in a constituent database.

Drop Calculation

Delete a stored calculation from a database. Minimum permission required: Database Manager.

Syntax

```
▶▶ drop calculation CALC-NAME ▶▶
```

You can delete calculations using **drop calculation**.

Keyword

Description

drop calculation <calc-name> Delete the specified calculation.

Example

```
drop calculation Sample.basic.calcname;
```

Deletes a calculation from Sample.basic.

Drop Database

Delete a database from the system. Minimum permission required: Database Manager. If the database has outstanding locks, clear them first, or use **force** to drop with locks.

Syntax

```
▶▶ drop database DBS-NAME [ force ] ▶▶
```

You can delete databases using **drop database**.

Keyword	Description
----------------	--------------------

<code>force</code>	Delete a database that may have locked objects.
--------------------	-------------------------------------------------

Example

```
drop database Sample.Basic force;
```

Deletes the database Sample.Basic, even if client users have outstanding locks on Sample.Basic.

Drop Drillthrough

Delete a drill-through URL definition used to link to content hosted on Oracle ERP and EPM applications.

Syntax

```
▶▶— drop drillthrough URL-NAME —▶▶
```

Example

```
drop drillthrough sample.basic.myURL;
```

See Also

- [alter drillthrough](#)
- [create drillthrough](#)
- [display drillthrough](#)

Drop Filter

Delete a security filter from the database. Minimum permission required: Database Manager.

Syntax

```
▶▶— drop filter FILTER-NAME —▶▶
```

You can delete filters using **drop filter**.

Keyword	Description
----------------	--------------------

<code>drop filter <filter-name></code>	Delete a filter by name.
----------------------------------------------	--------------------------

Example

```
drop filter sample.basic.filter1;
```

Deletes the filter called filter1 from the sample.basic database.

Drop Function

Delete a custom-defined function from the system or from an application.

Minimum permission required: Application Manager to drop a local (application-level) function. Administrator to drop a global (system-level) function.

Syntax

```
▶▶ drop function FUNC-NAME ─────────────────▶▶
```

You can delete custom-defined functions using **drop function**.

Keyword	Description
---------	-------------

drop function <func-name>	Delete a custom-defined function by name.
---------------------------	-------------------------------------------

Notes

If you drop a custom-defined function after having associated it with an application (using [refresh custom definitions](#)), you may have to stop and restart the application for the drop to take effect.

Example

```
drop function sample.'@COVARIANCE';
```

Deletes the function called @COVARIANCE from the Sample application.

Drop Group

Delete a user group from the system. Users belonging to the group are not deleted.

An Essbase group can be deleted in two ways:

1. From Essbase using MaxL or Administration Services. To do this, you must be an Essbase administrator.
2. From Shared Services. To do this, you must be a Shared Services administrator.

Syntax

```
▶▶ drop group GROUP-NAME ─────────────────▶▶  
└── from security_file ───┘
```

You can delete security groups using **drop group**.

Keyword	Description
---------	-------------

drop group <group-name>	Delete a security group by name. Members of the group are not deleted, but their membership to the group becomes obsolete.
-------------------------	----------------------------------------------------------------------------------------------------------------------------

from security_file	When Essbase is in EPM System security mode, you can use this syntax to remove the user from the Essbase security file, without de-provisioning the user from Shared Services. Calculation and filter associations also are removed.
--------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Example

```
drop group big_group;
```

Deletes the group called big_group from the system.

Drop Location Alias

Delete from the database a location alias identifying a host name, application, database, user name, and password. Minimum permission required: Database Manager.

Syntax

```
▶▶ drop location alias LOCATION-ALIAS-NAME ◀◀
```

You can delete location aliases using **drop location alias**.

Keyword

Description

drop location alias <location-alias-name> Delete a location-alias definition.

Example

```
drop location alias Main.Sales.EasternDB;
```

Drops the location alias called EasternDB in the Main.Sales database.

Drop Lock

Remove locks acquired through a spreadsheet operation.

Note: Data locks do not apply to aggregate storage applications.

Syntax

```
▶▶ drop lock [ on system | on application APP-NAME | on database DBS-NAME | all ] [ all | held by USER-NAME ] ◀◀
```

Keyword

Description

drop lock on system all

Drops all locks by all users, for all databases on the system.

drop lock all

Same as "drop lock on system all"

drop lock on system

Same as "drop lock on system all"

drop lock

Same as "drop lock on system all"

Keyword	Description
drop lock on application APP-NAME	Drops all locks on the application, for all users.
drop lock on application APP-NAME held by USER-NAME	Drops locks on the application which are held by a specific user.
drop lock on database DBS-NAME	Drops all locks on the database, for all users.
drop lock on database DBS-NAME held by USER-NAME	Drops locks on the database which are held by a specific user.
drop lock held by USER-NAME	Drops all locks held by a specific user, on any application or database.

Drop Macro

Delete a custom-defined macro from the system or from an application.

Minimum permission required: Application Manager to drop a local (application-level) macro.
Administrator to drop a global (system-level) macro.

Syntax

▶▶ drop macro **MACRO-NAME** ◀◀

You can delete custom-defined macros using **drop macro**.

Keyword	Description
drop macro <macro-name>	Delete a custom-defined macro.

Notes

If you drop a custom-defined macro after having associated it with an application (using [refresh custom definitions](#)), you may have to stop and restart the application for the drop to take effect.

Example

```
drop macro sample.'@COVARIANCE';
```

Deletes the macro called @COVARIANCE from the Sample application.

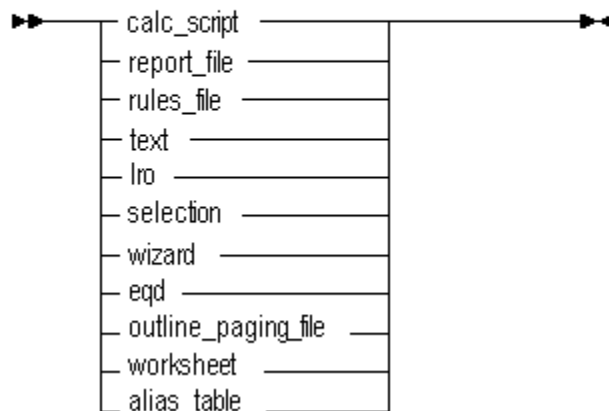
Drop Object

Remove database-related file objects stored in database directories.

Syntax

►► drop object **OBJ-NAME** of type <OBJ-TYPE> [force] ►►

<OBJ-TYPE> ::=



Keyword Description

...force If the object is locked by a user or process, unlock it and delete it.

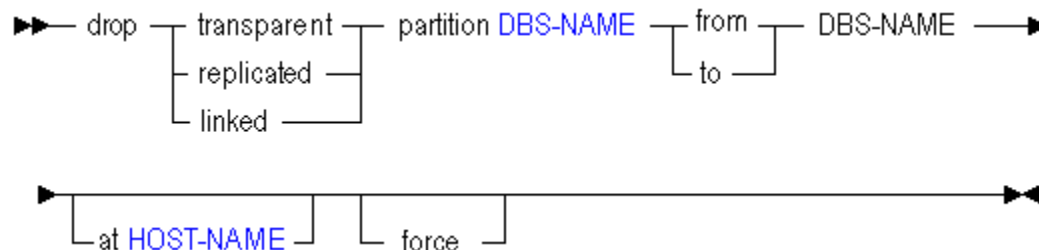
Notes

To drop a partition, use [drop partition](#).

Drop Partition

Delete from the system a partition definition between two databases. Database Manager permission for each database is required.

Syntax



You can delete partition definitions in the following ways using **drop partition**.

Keyword

Description

- | | |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| drop...partition...from | Remove a transparent, replicated, or linked partition definition between the current target database and a source database. |
| drop...partition...to | Remove a transparent, replicated, or linked partition definition between the current source database and a target database. |
| at <host-name> | Optionally specify the host computer name, if removing a partition definition associated with a remote server. The host name can be an IP address; for example, '127.0.0.1'. |

Keyword	Description
force	Specify that the source half of a partition definition should be dropped regardless of whether the target half is missing or invalid. For more information, see “Forcing Deletion of Partitions” on page 925 .

Notes

If the **create partition** statement used was of the format:

```
create partition SOURCE to TARGET;
```

Then the only permutations of the **drop partition** statement that will have effect are:

```
drop partition SOURCE to TARGET;
drop partition TARGET from SOURCE;
```

Example

```
create or replace replicated partition sampeast.east area '@IDESCENDANTS("Eastern
Region"), @IDESCENDANTS(Qtrl)' to samppart.company at localhost;

drop replicated partition Samppart.Company from Sampeast.East;
```

Drop Trigger

Remove a trigger created to track state changes over a selected cube area. For more information about the Essbase triggers feature, see the *Oracle Essbase Database Administrator's Guide*.

Syntax

```
▶▶— drop trigger TRIGGER-NAME —▶▶
```

Example

```
drop trigger Sample.Basic.WatchCosts ;
```

Drop Trigger Spool

Delete the log file created by a trigger. Triggers track state changes over a selected cube area. For more information about the Essbase triggers feature, see [Defining Triggers](#).

Syntax

```
▶▶— drop trigger_spool [ SPOOL-NAME ]
                        [ all on database DBS-NAME ] —▶▶
```

Drop User

Delete a user account from the system.

Permission required: `create_user`, unless Essbase is in EPM System security mode.

In EPM System security mode, an Essbase user can be deleted in two ways:

1. From Essbase using MaxL or Administration Services. To do this, you must be an Essbase administrator. You must additionally have the Shared Services Directory Manager role.
2. From Shared Services. To do this, you must be a Shared Services administrator.

Syntax

```

▶▶ drop user USER-NAME
      └── from security_file ──▶▶
  
```

You can delete users using **drop user**.

Keyword	Description
---------	-------------

drop user <user-name>	Delete an Essbase user account by user name.
-----------------------	----------------------------------------------

from security_file	When Essbase is in EPM System security mode, you can use this syntax to remove the user from the Essbase security file, without de-provisioning the user from Shared Services. Calculation and filter associations also are removed.
--------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Example

```
drop user Fiona;
```

Deletes the user Fiona from the system.

Execute Calculation

[Click here for aggregate storage version](#)

Execute a stored calculation, the stored default calculation (determined by [alter database](#)), or an anonymous (non-stored) calculation string.

Minimum permissions required:

- For stored calculations (CALC-NAME): Granted access to the calculation.
- For anonymous calculations (CALC-STRING): Database Manager permission.
- For default calculation: execute privilege.

Syntax

```

▶▶ execute calculation
      ├── CALC-NAME ───────────▶▶
      ├── CALC-NAME on database DBS-STRING ───▶▶
      ├── CALC-STRING ─── on DBS-NAME ───▶▶
      └── default ───────────▶▶
  
```

You can run calculations in the following ways using **execute calculation**.

Keyword	Description
---------	-------------

execute calculation <calc-name>	Run the specified stored calculation script.
---------------------------------	----------------------------------------------

Keyword	Description
<calc-name> on database	Run the specified stored calculation script against the specified database.
<calc-string> on <dbs-name>	Run an anonymous calculation, whose body is contained in <calc-string>, against the specified database.
default on <dbs-name>	Run the default calculation against the specified database.

Notes

A stored calculation can be associated with an application/database, or with an application only. To execute a calculation stored at the application level, you must specify which database to calculate using the syntax 'on database STRING.'

Example

```
execute calculation Sample.Basic.calcname;
```

Calculates the Sample.Basic database using a stored calculation file associated with the database.

```
execute calculation Sample.calcname on database Basic;
```

Calculates the Sample.Basic database using a stored calculation file associated with the application Sample.

```
execute calculation
'SET MSG ERROR;
CALC ALL;'
on Sample.basic;
```

Calculates Sample.Basic using an anonymous (unstored) calculation string.

Execute Aggregate Process

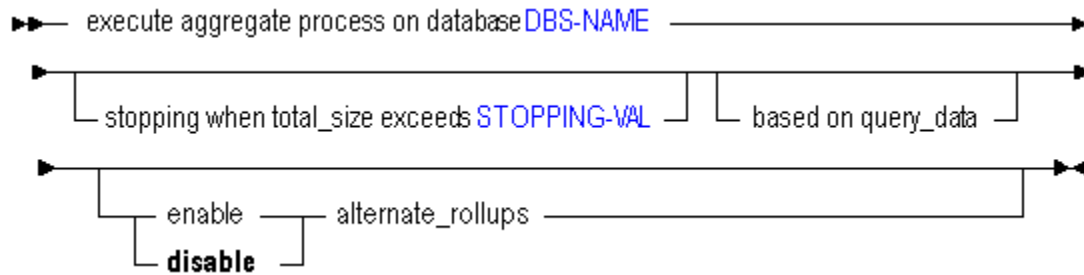
Perform an aggregation, optionally specifying the maximum disk space for the resulting files, and optionally basing the view selection on user querying patterns. This statement is only applicable to aggregate storage databases.

This statement causes Essbase to:

1. Select 0 or more aggregate views based on the stopping value and/or on querying patterns, if given.
2. Build the views that were selected.

For more information about aggregate views, see the *Oracle Essbase Database Administrator's Guide* and the *Oracle Essbase Administration Services Online Help*.

Syntax



You can aggregate an aggregate storage database in the following ways using `execute aggregate process`.

Keyword	Description
stopping when total_size exceeds...	Aggregate whichever views Essbase selects, with the exception that the maximum growth of the aggregated database must not exceed the given ratio.
based on query_data	Aggregate whichever views Essbase selects, based on collected user querying patterns. This option is only available if query tracking is turned on, using <code>alter database <db-name> enable query_tracking</code> .
enable disable alternate_rollups	If enabled, secondary hierarchies (with default level usage) are considered for view selection. Default: disabled (no secondary hierarchies are considered).

Notes

- View selection (step 1) can be performed independently of aggregation by using `execute aggregate selection`. Aggregation (step 2) can be performed without built-in view selection by using `execute aggregate build`.
- For small databases, the performance of building aggregate views in Essbase 9.3.1 and later versions may be slower than Essbase versions earlier than 9.3.1. However, Essbase 9.3.1 should perform better for databases larger than a few hundred million cells, especially on computers with more than two processors and where the `CALCPARALLEL` configuration setting has been chosen appropriately.

Example

```
execute aggregate process on database ASOSamp.Sample  
stopping when total_size exceeds 1.3;
```

Selects and builds an aggregation of the ASOSamp Sample database that permits the database to grow by no more than 30% as a result of the aggregation.

```
execute aggregate process on database ASOSamp.Sample based on query_data;
```

Selects and builds an aggregation of the ASOSamp Sample database, where the views that Essbase selects for aggregation are based on the most frequently queried areas of the database.

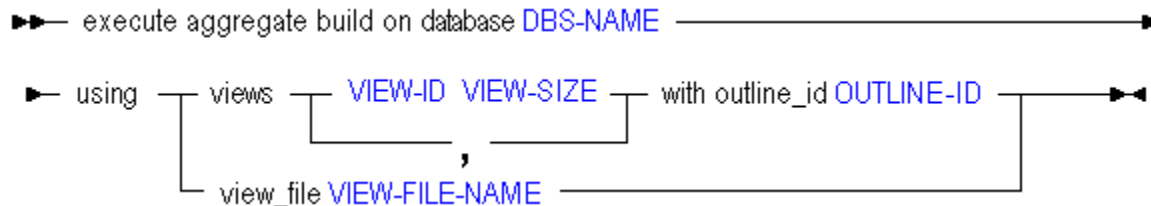
Execute Aggregate Build

Performs an aggregation based on the views selected by the `execute aggregate selection` statement.

The views to build must either be identified by their view IDs, obtained previously using [execute aggregate selection](#), or by a view selection saved in an aggregation script.

For more information about aggregate views, see the *Oracle Essbase Database Administrator's Guide* and the *Oracle Essbase Administration Services Online Help*.

Syntax



You can materialize aggregations in the following ways using `execute aggregate build`.

Keyword	Description
using views...	Builds an aggregation based on a previously selected view (or views) and the associated outline ID.
using view_file...	Builds an aggregation based on a saved view selection stored in an aggregation script. Omit the <code>.csc</code> file extension from the view file name when you issue the <code>execute aggregate build</code> statement.

Notes

- Although it is possible to pass arbitrary view-id and view-size arguments, this practice is not supported.
- Passing view-size arguments other than those returned by the `execute aggregate selection` command may cause unpredictable results.
- For small databases, the performance of building aggregate views in Essbase 9.3.1 and later versions may be slower than Essbase versions earlier than 9.3.1. However, Essbase 9.3.1 should perform better for databases larger than a few hundred million cells, especially on computers with more than two processors and where the `CALCPARALLEL` configuration setting has been chosen appropriately.

Example

```
execute aggregate build on database Sample.Basic using views 711 0.00375 with outline_ID  
4142187876;
```

Builds an aggregation of the Sample Basic database. The build is based on the view of an aggregate storage outline (identified as 4142187876) having the view ID 711, and a view size of 0.00375.

```
execute aggregate build on database Sample.Basic using view_file myView;
```

Builds an aggregation of the Sample Basic database based on the view saved in the aggregation script `myView.csc`.

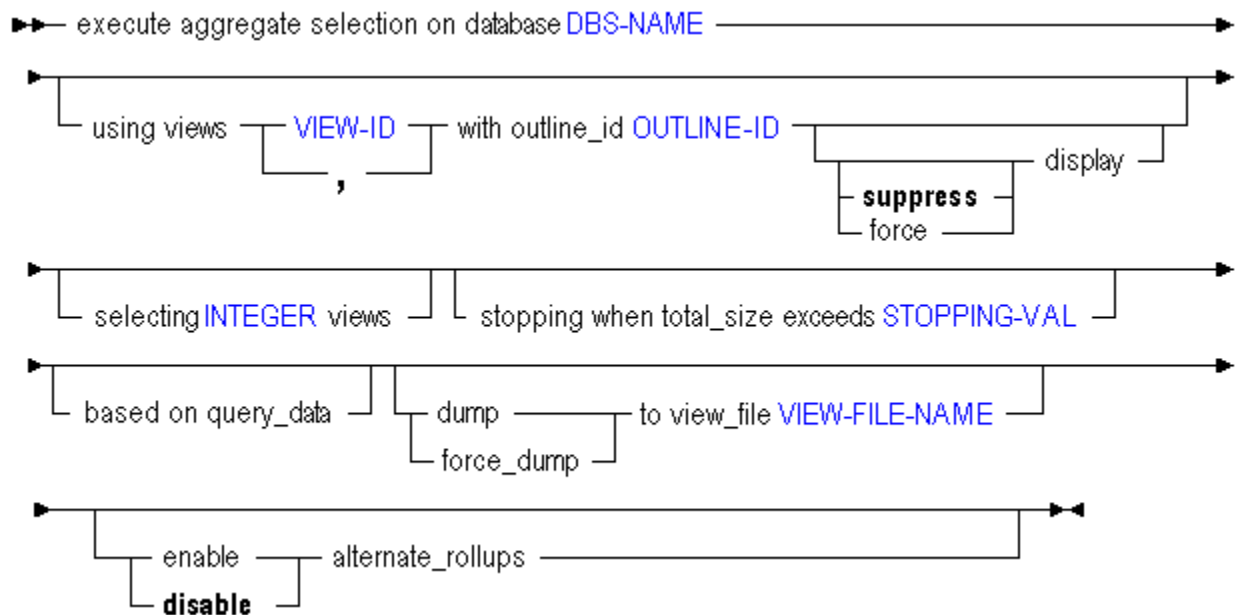
Execute Aggregate Selection

Select views of an aggregate storage database based on various selection criteria, and return the results in the form of a table or aggregation script. Next, use the tabular information or aggregation script to build an aggregation (materialize a view) using [execute aggregate build](#).

Note: View selection and aggregation can be performed by Essbase in a single step by using [execute aggregate process](#). However, the use of the two separate statements **execute aggregate selection** and **execute aggregate build** enables you more control of the selection criteria.

For more information about aggregate views, see the *Oracle Essbase Database Administrator's Guide* and the *Oracle Essbase Administration Services Online Help*.

Syntax



You can select views in the following ways using **execute aggregate selection**.

Keyword	Description
using views...with outline_ID	Selects views based on pre-selected view IDs. The view IDs are obtained from previous executions of the statement.
using views...with outline_ID...force display	Selects views based on pre-selected view IDs, including the pre-selected views IDs themselves.
using views...with outline_ID...suppress display	Selects views based on pre-selected view IDs, skipping the pre-selected views IDs themselves. This is the default behavior even if the suppress keyword is omitted.
selecting <INTEGER> views	Selects views up to a maximum number of views.

Keyword	Description
stopping when total_size exceeds	Selects views, specifying a storage stopping value in terms of a factor times the size of the unaggregated input (level 0) values. For example, a stopping value of 1.5 means that the view selection should permit the database to grow by no more than 50% as a result of the aggregation.
based on query_data	<p>Selects views based on previously collected query-tracking data. You must have enabled query tracking using <code>alter database <db-name> enable query_tracking</code>. After enabling query tracking, allow sufficient time to collect user data-retrieval patterns before performing an aggregate selection based on query data.</p> <p>Query tracking records information about every query executed on the database, so that it can be used as a basis for view selection. Query-based view selection helps to improve query performance when the distribution of user queries is skewed.</p> <p>For every level combination, the cost of retrieving cells is recorded. The recording continues until the application is shut down or until the recording is explicitly turned off using <code>alter database <db-name> disable query_tracking</code>. In both cases, all the query cost data is discarded, and the recording stops (and will not continue when the application starts again).</p> <p>All query cost data becomes invalid when additional views are built.</p>
dump to view_file	<p>Saves the view selection to an aggregation script. If the specified script name already exists, an error is returned. To overwrite an existing script, use the <code>force_dump</code> keyword.</p> <p>The aggregation script contains information derived during the aggregate view selection. You can materialize the aggregation at a different time by running the aggregation script. For example: <code>execute aggregate build on database <db-name> using view_file <view-file-name></code></p>
force_dump to view_file	Saves the view selection to an aggregation script. If the specified script name already exists, the <code>force_dump</code> keyword causes it to be overwritten.
enable disable alternate_rollups	If enabled, secondary hierarchies (with default level usage) are considered for view selection. Default: disabled (no secondary hierarchies are considered).

Example

```
execute aggregate selection on database ASOSamp.Sample;
```

Performs the default view selection for ASOSamp Sample. This statement selects the same views as `execute aggregate process on database ASOSamp.Sample` would build.

```
execute aggregate selection on database ASOSamp.Sample using views 711, 8941 with outline_ID 4142187876;
```

Selects views based on the pre-selected view IDs. The view IDs are obtained from previous executions of the statement.

```
execute aggregate selection on database ASOSamp.Sample using views 711, 8941 with outline_ID 4142187876 force display;
```

Selects views based on the pre-selected view IDs. `force display` is used to include the pre-selected views (711 and 8941) in the new selection.

```
execute aggregate selection on database ASOSamp.Sample selecting 9 views;
```

Selects a maximum of nine views of ASOSamp Sample.

```
execute aggregate selection on database ASOSamp.Sample stopping when total_size exceeds 1.2;
```

Selects an aggregation of the ASOSamp Sample database that, when built, would permit the database to grow by no more than 20% as a result of the aggregation.

```
execute aggregate selection on database ASOSamp.Sample based on query_data;
```

Selects views based on previously collected query-tracking data. You must have enabled query tracking using `alter database <db-name> enable query_tracking`.

```
execute aggregate selection on database ASOSamp.Sample
dump to view_file myView;
```

Selects a default aggregation of the ASOSamp Sample database, saving the selection to `APP \DB\myView.csc`. You can materialize the view later by running the aggregation script `myView.csc`. For example:

```
execute aggregate build on database ASOSamp.Sample using view_file 'myView.csc';
```

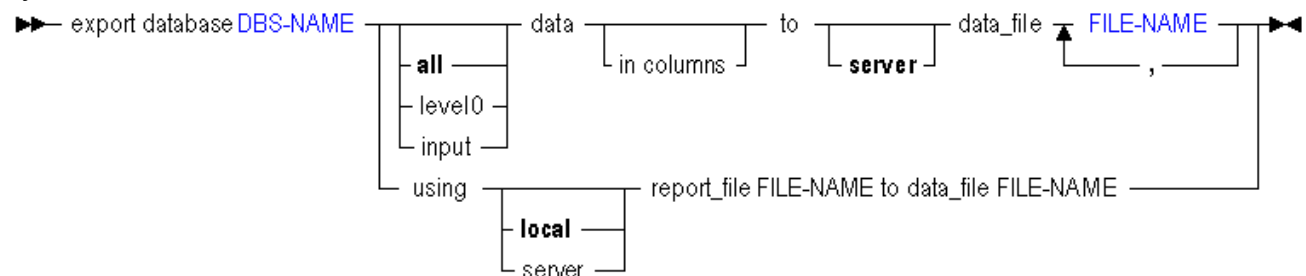
Export Data

[Click here for aggregate storage version](#)

Export all data, level-0 data, or input-level data, which does not include calculated values. Export files are stored in the `ARBORPATH/app` directory on the server unless an absolute path is specified. To use Report Writer, export the data using a report file.

Minimum permission required: Read. This statement requires the database to be started.

Syntax



You can export data from a database in the following ways using `export data`.

Keyword	Description
export database <db-name> all data...	Export all data in the specified database to the <code>\$ARBORPATH/app</code> directory on the server. Note: Exporting data does not clear the data from the database.
export database <db-name> level0 data...	Export level-0 data blocks only (blocks containing only level-0 sparse member combinations. Note that these blocks may contain data for upper level dense dimension members.) A level-0 block is created for sparse member combinations when all of the members of the sparse combination are at the bottom of dimension branches. Note: Exporting data does not clear the data from the database.
export database <db-name> input data...	Export only blocks of data where the block contains at least one data value that was loaded (imported), rather than created as the result of a calculation.

Keyword	Description
export database <dbs-name> ... data in columns	Export data in columns, to facilitate loading the exported data into a relational database. In each row, the columnar format displays a member name from every dimension. Names can be repeated from row to row. Columnar format provides a structure to the exported data, so that it can be used for further data processing by applications other than Essbase tools. In non-columnar format, sparse members identifying a data block are included only once for the block. Because the export file in non-columnar format is smaller than in columnar format, reloading a file in non-columnar format is faster.
export database <dbs-name> ...using...report_file...	Run a stored report script, exporting a subset of the database.

Notes

- To export data in parallel, specify a comma-separated list of export files. The number of threads Essbase uses depends on the number of file names you specify.

If the data for a thread exceeds 2 GB, Essbase may divide the export data into multiple files with numbers appended to the file names.

The naming convention for additional export files is as follows: _1, _2, etc. are appended to the additional file names. If the specified output file name contains a period, the numbers are appended before the period. Otherwise, they are appended at the end of the file name.

For example, if the given file name is /home/exportfile.txt, the next additional file is /home/exportfile_1.txt. If the file name is /home/exportfile, the next additional file is /home/exportfile_1.
- To export data in column format, use the optional "in columns" grammar.
- During a data export, the export process allows users to connect and perform read-only operations.
- When MaxL exports data from a Unicode-mode application, the export file is encoded in UTF-8. You cannot use UTF-8-encoded export files from a Unicode-mode application to import data to a non-Unicode-mode application. For more information about file encoding, see the Unicode section of the *Oracle Essbase Database Administrator's Guide*.
- MaxL cannot export databases with names containing hyphens (-). To export databases with names containing hyphens, use Administration Services.

Example

Example 1 (Export Data)

```
export database sample.basic data to data_file 'D:\\fileout', 'D:\\fileout2', 'D:\\fileout3';
```

Exports data concurrently to a list of file names.

Example 2 (Export Data)

```
export database sample.basic input data
to data_file 'exp_input.exp';
```



```
export database sample.basic using report_file '$ARBORPATH/App/Sample/Basic/async.rep'
to data_file 'home/month2.rpt';
```

Note: In the path to the report file in the above UNIX example, double quotation marks are used to allow variable expansion in the single-token FILE-NAME, and single quotation marks are required because there are special characters (see “MaxL Syntax Notes” on page 768) in the file name.

```
export database sample.basic using report_file 'EssbaseServer\\App\\Sample\\Basic\\
\\async.rep' to data_file 'c:\\home\\month2.rpt';
```

Note: In the file paths in the above Windows example, single quotation marks are required because there are special characters (see “MaxL Syntax Notes” on page 768) in the file name. Two backslashes (\\) are required by the MaxL Shell to indicate one backslash, because the backslash has a special meaning to the MaxL Shell.

Export LRO

Export linked-reporting-object information, and binary files if the database has file-type LROs, to a directory on the Essbase Server computer.

Syntax

```
▶▶ export database DBS-NAME lro to  directory  ▶▶
```

You can export LRO information from a database in the following ways using `export lro`.

Keyword	Description
to server directory	Export the LRO information to a directory you specify on the Essbase Server to which you are connected.
to local directory	Export the LRO information to a directory you specify on the current computer.

Notes

- This statement requires the database to be started.
- MaxL creates exactly one export directory; it does not create a directory *structure*. For example, if `c:\temp` exists, MaxL will create `c:\temp\exports`, but not `c:\temp\exports\to\this\long\path`.
- If the specified export directory already exists, the export LRO statement will fail. This is a safeguard against overwriting existing export directories.
- If you do not specify a *full path* for an export directory to be created on the client or server, MaxL uses your short directory specification (“DBS-EXPORT-DIR” on page 779) as a suffix, and creates the destination export-directory in the `ARBORPATH\app` directory with

a prefix of `appname-dbname-`. If you do specify a full path, MaxL creates whatever directory you specify.

It is recommended that you create export directories in the application/ database directory, as MaxL can only display or delete export directories that are in the application/database directory.

- When MaxL exports LROs from a database, if the database is from a Unicode-mode application, the exported LRO-catalog file is encoded in UTF-8. You cannot use UTF-8-encoded export files from a Unicode-mode application to import LROs to a non-Unicode mode application. For more information about file encoding, see the Unicode section of the *Oracle Essbase Database Administrator's Guide*.

Example

```
export database sample.basic lro to server directory '../home/temp/lros';
```

Exports LRO-catalog information, and binary files if the database has file-type LROs, to a server directory called `home/temp/lros`. The directory contains file-type LROs, if applicable, and the LRO-catalog export file `lros.exp`. These can be brought back into a database using [import lro](#).

```
export database sample.basic lro to server directory 'exportedLROs';
```

Exports LRO-catalog information, and binary files if the database has file-type LROs, to a server directory `$ARBORPATH/app/sample-basic-exportedLROs`. The directory contains file-type LROs, if applicable, and the LRO-catalog export file named `sample-basic-exportedLROs.exp`. These can be brought back into a database using [import lro](#).

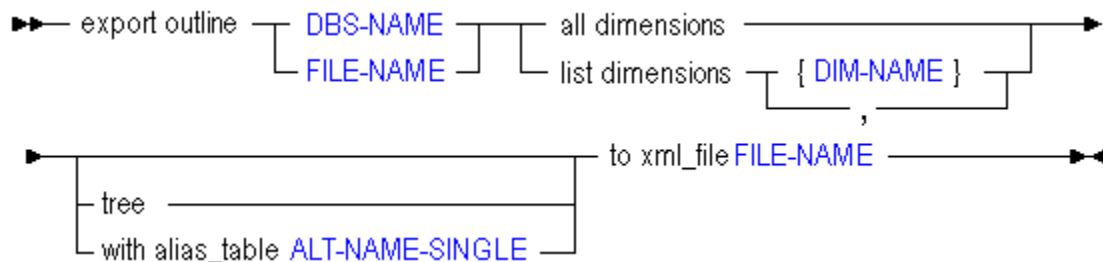
```
export database sample.basic lro to server directory 'D:\\MaxL\\LROexports\\dir';
```

On Windows, exports LRO-catalog information to a new directory `dir` under the existing directory structure `D:\MaxL\LROexports`. The double backslashes (`\\`) must be used because a single backslash is an escape character to MaxL.

Export Outline

Export metadata, either from the active database outline or an input outline file, to a specified XML file. Permission required: database manager.

Syntax



You can export metadata information from a database in the following ways using `export outline`.

Keyword	Description
DBS-NAME	Specify the database name instead of the outline file path.
FILE-NAME	Specify the outline file path instead of the database name.
all dimensions	Export information about all dimensions in the database.
list dimensions	Export information about only the listed dimensions. Specify each dimension name within curly braces, and separated by commas.
tree	Export only the member names in the hierarchy, omitting full metadata details.
with alias_table	Export using only the member names indicated in the specified alias table.
to xml_file	Specify the full path to the output XML file.

Notes

- This statement requires the database to be started.
- The following general outline information is included in the XML export:
 - Case sensitiveness
 - Outline Type
 - Duplicate Member Names allowed
 - Typed Measures Enabled
 - Date Format
 - Varying Attributes Enabled
 - Alias Table count and list
 - Active Alias Table
 - Attribute information
 - Auto configure
 - Text list definitions
 - Universal member comments
 - Locale, if it exists
 - Query hint list (if aggregate storage)
 - Get Implied Shared Setting
- The following dimension information is included in the XML export:
 - Name
 - Two pass calc
 - Type
 - Text list, if text typed
 - Has relational descendants

- IsHAEnabled (Hybrid Analysis enabled)?
- Formula
- Format String
- Comment
- Extended member comment
- Dimension category
- Attribute type
- Data Storage
- Dimension Storage
- Alias Names, if any
- UDAs, if any
- Consolidation
- Attribute dimension associated
- Independent dimensions, if any
- Time balance
- Skip options
- Variance reporting
- Currency conversion
- Currency conversion member
- Dynamic Time Series enabled list
- Attachment level, if linked attribute dimension
- Dimension solve order
- Is Non Unique dimension?
- Hierarchy type
- Level usage for aggregation (for aggregate storage hierarchies)
- Is Compression dimension? (if aggregate storage)
- Storage category
- The following member information is included in the XML export:
 - Name
 - Two pass calc
 - Type
 - Text list, if text typed
 - Is shared?
 - Shared member name, if shared
 - Formula

- Format string
- Comment
- Extended member comment
- Attribute type
- Data storage
- Dimension storage
- Alias names, if any
- UDAs, if any
- Consolidation
- Attribute member associated
- Validity sets, if any
- Time balance
- Skip options
- Variance reporting
- Currency conversion
- Currency conversion member
- Member solve order (if aggregate storage)
- Level usage for aggregation (for aggregate storage hierarchy members)

Example

```
export outline sample.basic all dimensions to xml_file "c:/temp/basic.xml";
```

Exports all outline information from Sample Basic to the specified XML file, `basic.xml`.

```
export outline sample.basic list dimensions {"Product", "Market"} tree to xml_file "c:/temp/basic.xml";
```

Exports information about Sample Basic dimensions Product and Market from to the XML file.

```
Export outline "c:/temp/basic.otl" all dimensions with alias_table "Default" to xml_file "c:/temp/basic.xml";
```

Exports information about all dimensions in Sample Basic from the specified outline file to the XML file, using only default alias names.

Export Security File

Writes the contents of the Essbase security file (`essbase.sec`) to a readable, text file (ASCII format) on the system where Essbase Server resides. The statement is run against the Essbase Server instance for which you are currently logged in. The Essbase Server instance can be one that is run as a service.

Exporting the contents of the Essbase security file is useful when you want to review the security information for an Essbase Server instance. Be sure to follow your company's security procedures to ensure the integrity of the data.

Required permission: Essbase Administrator.

Syntax

```
▶▶ export security_file to data_file FILE-NAME ◀◀
```

Notes

- *FILE-NAME* specifies the name, including the path, of the text file to which the exported information is written. The path must be to a location on the system where Essbase Server resides. The file cannot be written to a client system. If a path is not specified, the text file is created in the *ARBORPATH\bin* directory.
- Running the `export security_file` statement against a pre-9.3.1 Essbase Server instance is not supported.
- The `export security_file` MaxL statement is similar to the DUMP agent command, except that the DUMP command cannot be run against an Essbase Server running as a service.

Example

```
export security_file to data_file essbase_security_file.txt;
```

Writes security information to a file named `essbase_security_file.txt` in the *ARBORPATH\bin* directory on the server system.

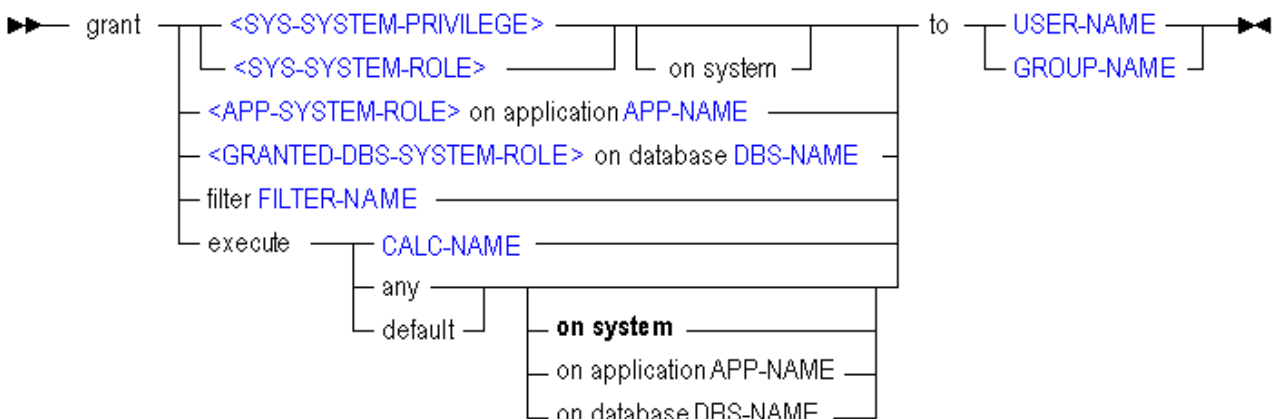
```
export security_file to data_file C:\security_review\essbase_security_file.txt;
```

Writes security information to a file named `essbase_security_file.txt` in the specified directory on the server system (`C:\security_review`).

Grant

Grant a permission, a filter or a stored calculation to a user or a group.

Syntax



You can grant permissions to users and groups in the following ways using **grant**.

Keyword	Description
create_application to...	Grant Create/Delete Applications permission to a user or group.
create_user to...	Grant Create/Delete Users/Groups permission to a user or group.
no_access to...	Revoke any permissions the user or group may have.
administrator to...	Grant Administrator permission to a user or group.
no_access on application...to...	Revoke any permissions the user or group may have on the specified application.
manager on application...to...	Grant Application Manager permission to a user or group for the specified application.
no_access on database...to...	Revoke any permissions the user or group may have on the specified database.
read on database...to...	Grant Read permission to a user or group for the specified database.
write on database...to...	Grant Write permission to a user or group for the specified database.
manager on database...to...	Grant Database Manager permission to a user or group for the specified database.
filter <filter-name> to...	Assign a filter to a user or group that grants or denies permissions to the specified database at a data-value level of detail.
execute <calc-name> to...	Grant the user or group permission to run the specified stored calculation script.
execute any on system to...	Grant the user or group permission to run any calculation against any database on the Essbase Server.
execute any on application...to...	Grant the user or group permission to run any calculation against any databases in the specified application.
execute any on database...to...	Grant the user or group permission to run any calculation against the specified database.
execute default on system to...	Grant the user or group permission to run the default calculation against any database on the Essbase Server.
execute default on application...to...	Grant the user or group permission to run the default calculation against any databases in the specified application.
execute default on database...to...	Grant the user or group permission to run the default calculation against the specified database. The default calculation is typically 'CALC ALL;', but it can be changed using alter application set default calculation .

Notes

Granting permissions:

At each level (system, application or database) existing roles are replaced. However, the built-in privileges `create_user` and `create_application` are not replaced.

After granting a permission to a user or group, it can be revoked by subsequently granting `no_access`. However, to prevent users from being able to load the application, you should also grant `no_access` at the application level.

Granting filters:

There may be only one filter per user per database. Therefore, granting a filter replaces any filters the user may already have on that database.

Filter permission can be revoked from users and groups by using the **revoke filter** clause of [Alter User](#) and [Alter Group](#).

Granting calculations:

A user or group may have any number of calculations per database. Therefore, granting a calculation adds it to the user or group's list of calculations. **Grant execute any** gives the user or group permission to execute all calculations, including the default calculation.

After granting execute permission, the permission can be revoked by subsequently granting `no_access` to the database. However, to prevent users from being able to load the application, you should also grant `no_access` at the application level.

Example

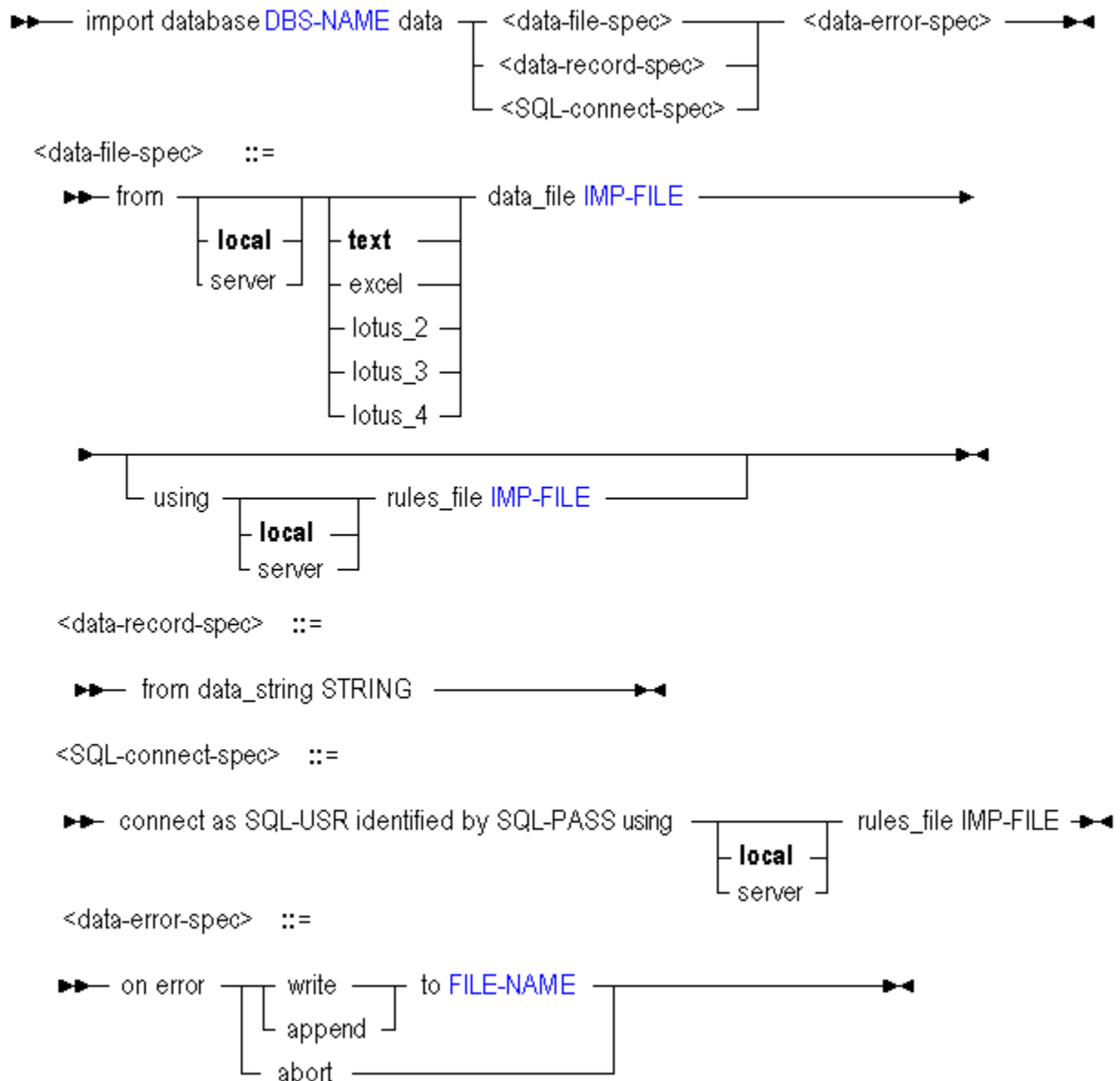
```
grant no_access to NewGroup;
grant administrator to Fiona;
grant manager on application Sample to Fiona;
grant read on database Sample.basic to Fiona;
grant filter Sample.basic.filter8 to Fiona;
```

Import Data

[Click here for aggregate storage version](#)

Import data from text or spreadsheet data files, with or without a rules file. Minimum permission required: Write.

Syntax



You can import data to a database in the following ways using **import data**.

Keyword	Description
import database <dbs- name> data from...	Specify whether the data import is from a local or server file, and what type of file to import data from.
...using ... rules_file	Import data into the database using a specified rules file.
...<data error spec> (on error...)	Required. Tell Essbase what to do in case of errors during the data load: abort the operation, or write or append to a specified error log.

Keyword	Description
----------------	--------------------

...<data record spec> from data_string	Load a single data record into the selected database.
----------------------------------------	-------------------------------------------------------

Example:

```
import database sample.basic data
from data_string
  "Sales" "COGS" "Marketing" "Payroll" "Misc" "Opening Inventory"
"Additions"
  "Ending Inventory" "100-10" "New York" "Jan" "Actual"
  678 271 94 51 0 2101 644 2067'
on error abort;
```

...<SQL connect spec> (connect as...)	If you are importing data from an SQL source, provide your SQL user name and password. You must always use a rules file when you load SQL data sources.
---------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------

Notes

- This statement requires the database to be started.
- When using the import statement, you must specify what should happen in case of an error.
- To import from a SQL data source, you must connect as the relational user name, and use a rules file.

Example

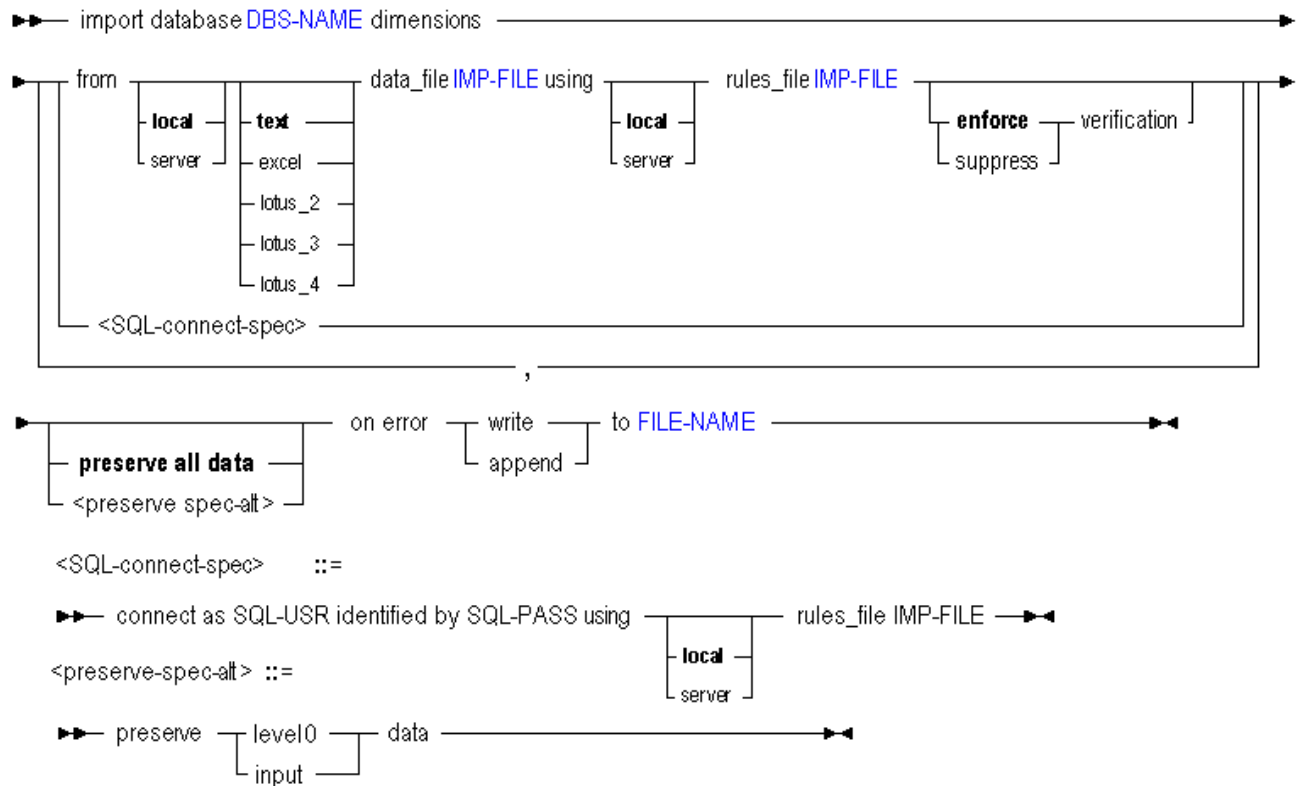
```
import database sample.basic data from data_file '$ARBORPATH\app\sample\basic\
\calcdat.txt' on error abort;
```

```
import database sample.basic data
from data_file '/data/calcdat.txt'
using rules_file '/data/rulesfile.rul'
on error write to '/logs/dimbuild.log';
```

Import Dimensions

Import dimensions from text or spreadsheet data files, using a rules file. Minimum permission required: Write.

Syntax



You can import dimensions to a database in the following ways using **import dimensions**.

Keyword	Description
import database <db-name> dimensions from...	Specify whether the dimension import is from a local or server file, and what type of file to import the dimension from.
...using ... rules_file	Import dimensions into the database outline using a specified rules file.
...enforce verification	Verify the outline resulting from the dimension build. This is the default behavior.
...suppress verification	Do not verify the outline resulting from the dimension build.
Caution! Using this option defers restructuring.	
...preserve all data	If you need to preserve all data when importing dimensions, specify that here.
...on error...	Tell Essbase what to do in case of errors during the dimension build: abort the operation, or write or append to an error log.
...<SQL connect spec> (connect as...)	If you are importing dimensions from an SQL source, provide your SQL user name and password. You must always use a rules file when you load SQL data sources.
...<preserve spec alt> (preserve...data)	If you need to preserve level-0 or input data when importing dimensions, specify that here.

Notes

- This statement requires the database to be started.
- When using the import statement, you must specify how error logs should be handled.
- When multiple files are included in the same statement, restructure is deferred until all files have been processed. The deferred-restructure type of dimension build has been called an incremental dimension build.
- When the **suppress verification** option is used, restructure is deferred.
- When multiple files are included in the same statement, **be sure verification is enforced for the last file.**
- To import from a SQL data source, you must connect as the relational user name, and use a rules file.

Example

```
import database sample.basic dimensions
from data_file '/data/calcdat.txt'
using rules_file '/data/rulesfile.rul'
on error append to '/logs/dimbuild.log';
```

Deferred-Restructure Examples

For Data File Sources:

```
import database sample.basic dimensions
from server text data_file 'genref' using server rules_file 'genref' suppress
verification,
from server text data_file 'level' using server rules_file 'level' suppress
verification,
from server text data_file 'time' using server rules_file 'time'
preserve input data on error append to 'C:\Hyperion\products\eas\client\dataload.err';
```

For SQL Sources:

```
import database sample.basic dimensions
connect as 'username1' identified by 'password1' using server rules_file 'genref',
connect as 'username2' identified by 'password2' using server rules_file 'level',
connect as 'username3' identified by 'password3' using server rules_file 'time'
on error append to 'C:\Hyperion\products\eas\client\dataload.err';
```

For Data and SQL Sources:

```
import database sample.basic dimensions
from server text data_file 'genref' using server rules_file 'genref',
from server text data_file 'level' using server rules_file 'level',
connect as 'username1' identified by 'password1' using server rules_file 'genref',
connect as 'username2' identified by 'password2' using server rules_file 'genref'
on error append to 'C:\Hyperion\products\eas\client\dataload.err';
```

Import LRO

Import Linked Reporting Objects (LROs) from the specified output directory created by [export lro](#). The directory contains an ASCII `.exp` file containing LRO-catalog information, and LRO binary files (if the database from which LROs were exported contained file-type LROs).
Minimum permission required: Write.

Syntax

```
▶▶ import database DBS-NAME lro from  directory IMPORT-DIR ▶▶
```

You can import exported LRO information to a database using **import lro**.

Keyword	Description
<code>import database <db-name> lro...</code>	Import Linked Reporting Objects (LROs) from the specified export directory on the local computer or on a remote server where the Essbase Server resides.

Notes

- This statement requires the database to be started.
- The specified import directory must come from the results of the [export lro](#) operation. The exported LRO-catalog file contains a record of the LRO file locations, cell notes, or URL text, and database index locations to use for re-importing to the correct data blocks.
- In the paths in the second two examples, double quotation marks are used to allow variable expansion in the string `IMPORT-DIR`, and single quotation marks are required because there are special characters (see [“MaxL Syntax Notes” on page 768](#)) in the path name.

Example

Windows Example

```
import database sample.basic lro
from server directory 'C:\Hyperion\products\Essbase\EssbaseServer\app\sample-
basic-lros';
```

```
import database sample.basic lro
from directory '$ARBORPATH\app\sample-basic-lros';
```

UNIX Example

```
import database sample.basic lro

from server directory '$ARBORPATH/app/sample-basic-lros';
```

From the subdirectory created by **export lro** in the `app` directory on the server, both the Windows and UNIX example statements above re-import the LRO-catalog information (and file-type LROs if applicable) that were exported to that location.

Query Archive_File

Retrieve information about the database backup archive file.

Minimum permission required: Read.

The database must be running.

Syntax

```
query archive_file FILE-NAME { get overview | list disk volume }
```

You can query archive file information using keywords.

Keyword	Description
get overview	Retrieve the following overview information: <ul style="list-style-type: none">● Application name● Database name● Time when the archive was performed
list disk volume	Retrieve a list of disk volume names. <p>On Windows, Essbase adds the default ARBORPATH drive (for example, the C : drive) as a disk volume, even if the database that you backed up does not store data on that disk volume.</p>

Example

```
query archive_file /Hyperion/samplebasic.arc get overview;
```

Retrieves overview information about the `samplebasic.arc` backup archive file.

```
query archive_file /Hyperion/samplebasic.arc list disk volume;
```

Retrieves disk volume information about the `samplebasic.arc` backup archive file.

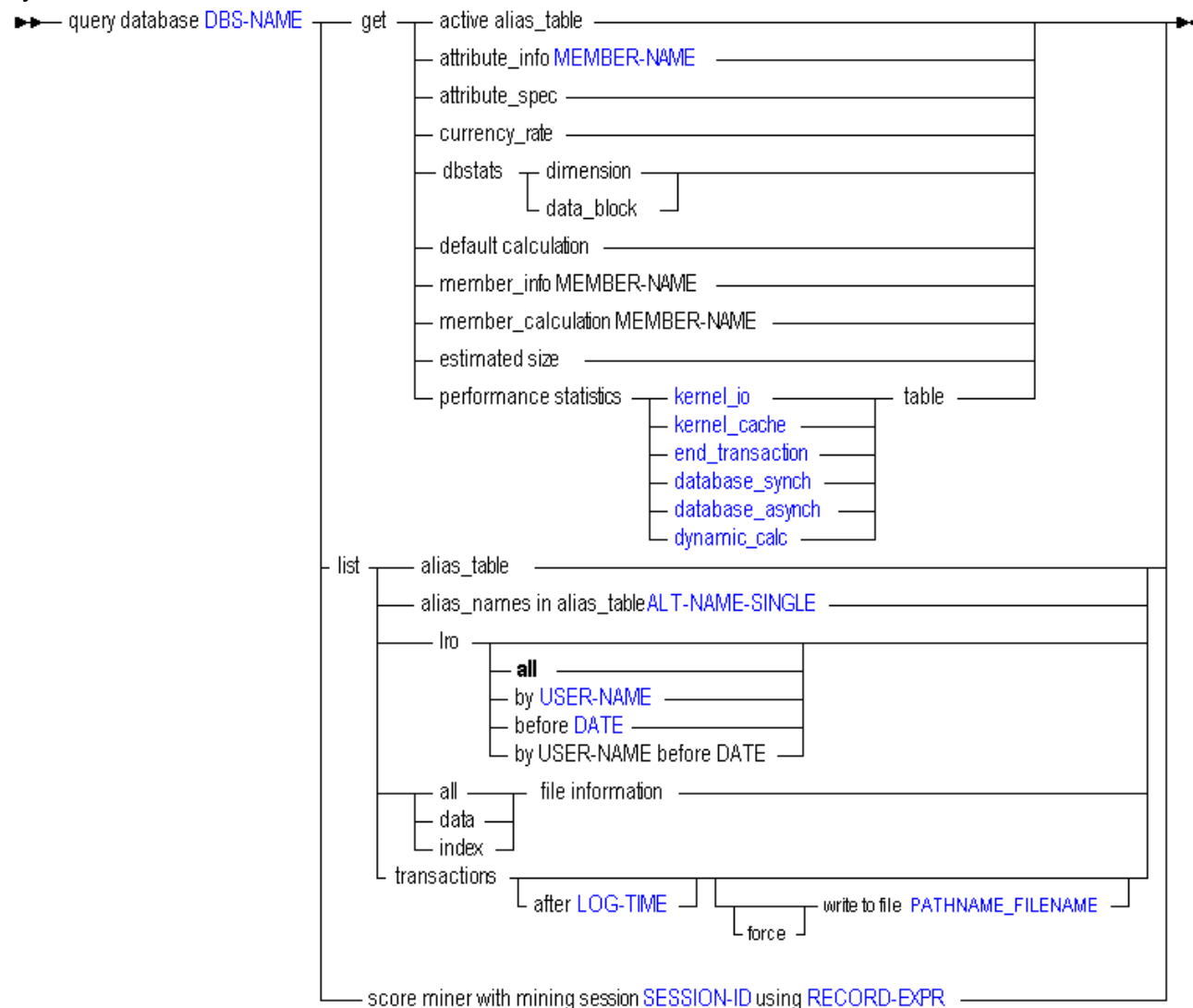
Query Database

[Click here for aggregate storage version](#)

Get advanced information about the current state of the database.

Minimum permission required: Read. This statement requires the database to be started.

Syntax



You can query for database information in the following ways using **query database**.

Keyword	Description
get active alias_table	Display the active alias table for the user issuing the statement.
get attribute_info	Get attribute member, dimension, and name information for the specified attribute member.
get attribute_spec	Display the current attribute specifications for the database. These specifications include attribute member name format, Attribute Calculation dimension member names, Boolean and date member names, and numeric range specifications. These settings are defined in Outline Editor.
get currency_rate	Display the currency rate for every currency partition.

Keyword	Description						
get dbstats dimension	<p>Get information about dimensions.</p> <p>Output</p> <p>The <code>index_type</code> field values are numeric, and translate as follows:</p> <table border="0"> <tr> <td>0</td> <td>Dense</td> </tr> <tr> <td>1</td> <td>Sparse</td> </tr> <tr> <td>3</td> <td>None (database is aggregate storage)</td> </tr> </table>	0	Dense	1	Sparse	3	None (database is aggregate storage)
0	Dense						
1	Sparse						
3	None (database is aggregate storage)						
get dbstats data_block	<p>Get information about data blocks. The information returned has little relevance to aggregate storage databases.</p> <p>Output</p> <p>The type field values are numeric, and translate as follows:</p> <table border="0"> <tr> <td>0</td> <td>Array</td> </tr> <tr> <td>1</td> <td>AVL (or "B+ Tree")</td> </tr> </table>	0	Array	1	AVL (or "B+ Tree")		
0	Array						
1	AVL (or "B+ Tree")						
get default calculation	<p>View the contents of the calculation designated as default for the database. The default calculation refers to either the relations defined in the database outline (CALC ALL) or to the set of calculation strings defined as the default database calculation.</p>						

Keyword	Description																																																								
get member_info MEMBER-NAME	<p>Get information on a specific member.</p> <p>Output</p> <p>The unary_type field values are numeric, and translate as follows:</p> <table border="0"> <tr><td>0</td><td>Add</td></tr> <tr><td>1</td><td>Subtract</td></tr> <tr><td>2</td><td>Multiply</td></tr> <tr><td>3</td><td>Divide</td></tr> <tr><td>4</td><td>Percent</td></tr> <tr><td>5</td><td>NoRollUp</td></tr> </table> <p>The member_tag_type field values translate as follows:</p> <table border="0"> <tr><td>0</td><td>SkipNone</td></tr> <tr><td>16384</td><td>SkipMissing</td></tr> <tr><td>32768</td><td>SkipZero</td></tr> <tr><td>49152</td><td>SkipBoth</td></tr> <tr><td>1</td><td>BalFirst</td></tr> <tr><td>2</td><td>BalLast</td></tr> <tr><td>4</td><td>TwoPass</td></tr> <tr><td>8</td><td>Average</td></tr> <tr><td>64</td><td>Expense</td></tr> </table> <p>Variations are possible. The field value consists of one of the first four "skip" values plus any/all/none of the last five values. Some examples:</p> <table border="0"> <tr><td>0</td><td>SkipNone</td></tr> <tr><td>77</td><td>SkipNone, BalFirst, TwoPass, Average, Expense</td></tr> <tr><td>16385</td><td>SkipMissing and BalFirst</td></tr> </table> <p>The first four "skip" values are base values, and added to them are combinations of 1, 2, 4, 8, and 64.</p> <p>The status field values are hexadecimal, and translate as follows:</p> <table border="0"> <tr><td>0</td><td>Normal</td></tr> <tr><td>1</td><td>Never Share</td></tr> <tr><td>2</td><td>Label</td></tr> <tr><td>4</td><td>Refer Share</td></tr> <tr><td>8</td><td>Refer Share (with different name)</td></tr> <tr><td>16</td><td>Implicit share</td></tr> <tr><td>32</td><td>Virtual Member (stored)</td></tr> <tr><td>64</td><td>Virtual Member (not stored)</td></tr> <tr><td>2048</td><td>Attribute</td></tr> <tr><td>32768</td><td>Referred</td></tr> </table>	0	Add	1	Subtract	2	Multiply	3	Divide	4	Percent	5	NoRollUp	0	SkipNone	16384	SkipMissing	32768	SkipZero	49152	SkipBoth	1	BalFirst	2	BalLast	4	TwoPass	8	Average	64	Expense	0	SkipNone	77	SkipNone, BalFirst, TwoPass, Average, Expense	16385	SkipMissing and BalFirst	0	Normal	1	Never Share	2	Label	4	Refer Share	8	Refer Share (with different name)	16	Implicit share	32	Virtual Member (stored)	64	Virtual Member (not stored)	2048	Attribute	32768	Referred
0	Add																																																								
1	Subtract																																																								
2	Multiply																																																								
3	Divide																																																								
4	Percent																																																								
5	NoRollUp																																																								
0	SkipNone																																																								
16384	SkipMissing																																																								
32768	SkipZero																																																								
49152	SkipBoth																																																								
1	BalFirst																																																								
2	BalLast																																																								
4	TwoPass																																																								
8	Average																																																								
64	Expense																																																								
0	SkipNone																																																								
77	SkipNone, BalFirst, TwoPass, Average, Expense																																																								
16385	SkipMissing and BalFirst																																																								
0	Normal																																																								
1	Never Share																																																								
2	Label																																																								
4	Refer Share																																																								
8	Refer Share (with different name)																																																								
16	Implicit share																																																								
32	Virtual Member (stored)																																																								
64	Virtual Member (not stored)																																																								
2048	Attribute																																																								
32768	Referred																																																								
get member_calculation MEMBER-NAME	View the formula associated with the selected member.																																																								
get estimated size	Display an estimate of the number of blocks a database will create after full calculation (CALC ALL), based on the number of blocks that exist before calculation. The database can have all data loaded, or it can have a random sampling of data loaded. Outlines that contain sparse formulas of any type or top-down formulas are not supported. Results of the estimation on such databases may be invalid.																																																								

Keyword	Description
performance statistics...table	Display one of several choices of performance statistics tables . Before you can use this statement, you must enable performance statistics gathering, using alter database DBS-NAME set performance statistics enabled.
list alias_table	Get a list of alias tables that are defined for the database.
list alias_names in alias_table	List the alias names defined in an alias table. Alias tables contain sets of aliases for member names and are stored in the database outline. Use this grammar to see a list of alias names defined in the specified table.
list lro	Get information about linked objects, including the object type, name, and description, based on criteria you specify. If you specify both a user name and modification date, objects matching both criteria are listed. If you specify no user name or date, a list of all linked objects in the database is displayed.
list...file information	Get accurate index and data file information. Provides index and data file names, counts, sizes, and totals, and indicates whether or not each file is presently opened by Essbase. The file size information is accurate. Note that the file size information provided by the Windows operating system for index and data files that reside on NTFS volumes may not be accurate.
list transactions	Display, in the MaxL Shell window, database transactions that were logged after the time when the last replay request was originally executed or after the last restored backup's time (which ever occurred later).
list transactions after LOG-TIME	Display, in the MaxL Shell window, database transactions that were logged after the specified time. Enclose the TIME value in quotation marks; for example: '11_20_2007:12:20:00'
list transactions after LOG-TIME write to file PATHNAME_FILENAME	Write the list of database transactions to the specified file. The list output is written to a comma-separated file on the Essbase Server computer. Provide the full pathname to an existing directory and the name of the output file. If only the output filename is provided, Essbase writes the file to the <i>ARBORPATH/app</i> directory. When writing to an output file that already exists, you must use the force grammar to overwrite the file.
list transactions force write to file PATHNAME_FILENAME	Overwrite the contents of an existing output file.
list transactions after TIME...write to file PATHNAME_FILENAME	Write the list of database transactions that were logged after the specified time to the specified file.
score miner ...	Scoring a model is similar to applying a model to the data in a database. However, scoring is executed synchronously and the results are not written back into the database; rather, they are returned in XML for Analysis format. To load a model in preparation for scoring, see create mining result . To score a model, use this statement. To unload the model after scoring, use alter system stop mining session.

Example

Example 1

```
query database Sample.Basic list transactions;
```

Displays, in the MaxL Shell window, Sample.Basic database transactions that were logged after the time when the last replay request was originally executed or after the last restored backup's time (which ever occurred later).

Example 2

```
query database Sample.Basic list transactions after '11_20_2007:12:20:00'  
write to file 'C:\\Hyperion\\products\\Essbase\\EssbaseServer\\app\\Sample\\Basic\\  
listoutput.csv';
```

Writes the transactions in the Sample.Basic database that were logged after November 20, 2007 at 12:20:00 to a CSV file in the Sample.Basic database directory.

Example 3

```
query database sample.basic get member_calculation 'Profit per Ounce';
```

Displays the formula associated with the 'Profit per Ounce' member.

Example 4

```
query database sample.basic list lro before '06_16_2008';
```

Displays information about linked objects, in the Sample.Basic database, that were modified before the specified time.

Refresh Custom Definitions

Refresh the definitions of custom-defined functions or macros associated with an application, without restarting the application.

Syntax

►► — refresh custom definitions on application **APP-NAME** — ◄◄

You can update Anylitic Services' record of custom-defined function and macro definitions using **refresh custom definitions**.

Keyword	Description
refresh custom definitions on application...	Refresh the definitions of custom-defined functions or macros associated with the specified application, without restarting the application. To refresh global definitions, issue the statement separately for each application on the Essbase Server.

Notes

- This statement re-reads the custom-defined function and macro records on the Agent, and associates newly created functions or macros with the specified application (since the last refresh, or since the last time the application was restarted).
- A local function or macro must have been created using the double naming convention to indicate application context: see [create function](#) or [create macro](#) for details.
- Invalidly defined functions and macros are not loaded to the application.

- Validation occurs at the application level only, during the refresh (not during creation). There is no validation on the system level.

Example

```
refresh custom definitions on application Sample;
```

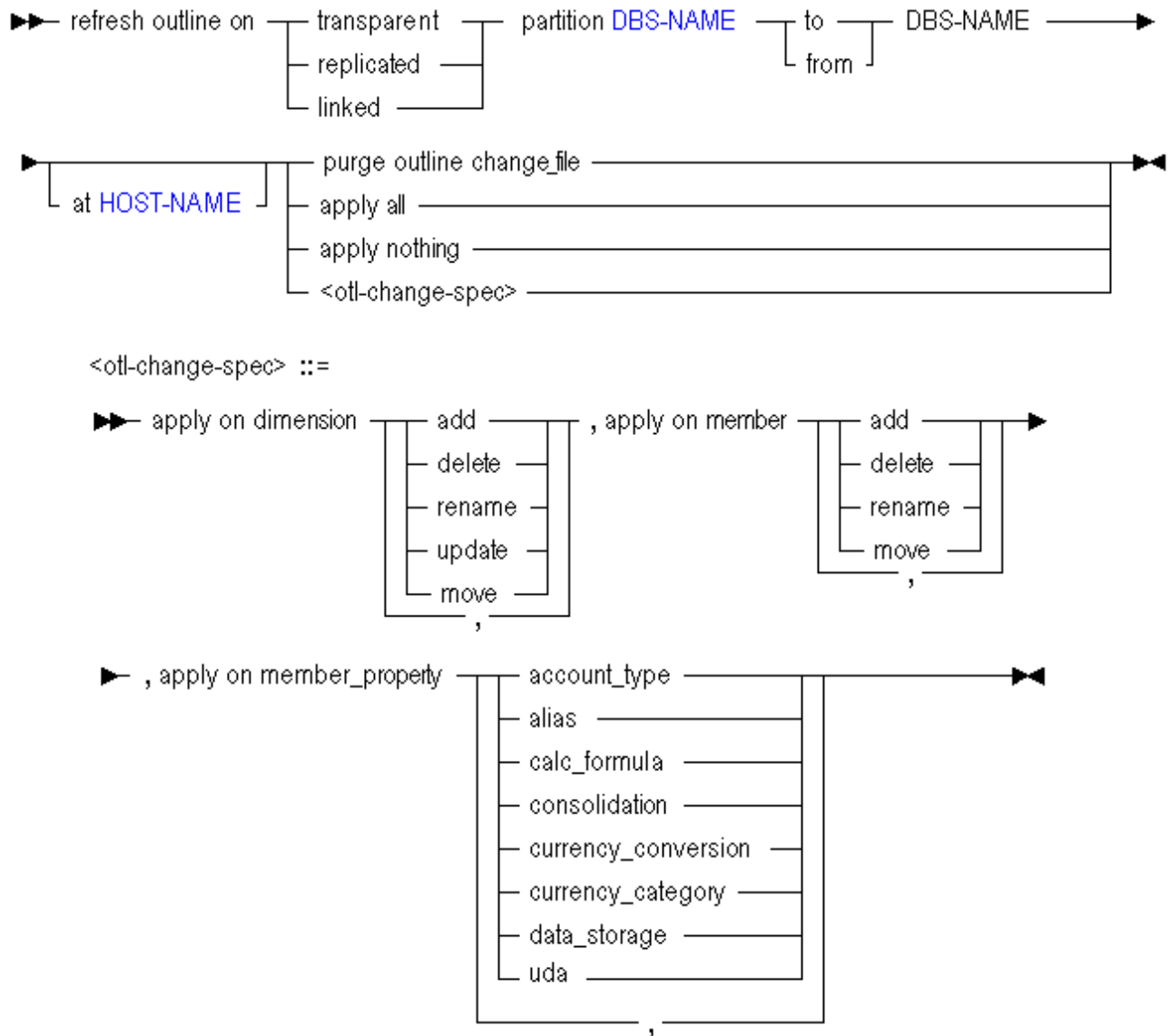
Loads all valid, newly created local functions and macros for the application Sample.

Refresh Outline

Synchronize the outlines between partitioned databases. Use this in the event that one outline has undergone changes to dimensions, members, or member properties, and you wish to propagate those changes to the partitioned database.

Outline synchronization is not currently enabled for partitions that involve aggregate storage databases.

Syntax



You can synchronize the outlines between partitioned databases using **refresh outline**.

Keyword	Description
...to...	Use the current source outline to refresh the remote target outline.
...from...	Refresh the current target outline using the remote source outline.
purge outline change_file	Clear any source outline changes that have already been applied to the target outline or have been rejected. Source outline changes that have not been applied or rejected are not deleted from the outline change file.
apply all	Refresh all aspects of the target outline, including dimension changes, member changes, and member property changes made to the source outline. This is the recommended method for refreshing outlines, because if you choose to omit some changes, those changes cannot be applied later.

Keyword	Description
apply nothing	Do not apply source outline changes to any aspects of the target outline. The target outline will be considered synchronized to the source, and the timestamp will be updated, although source changes were not actually applied to the target.
apply on dimension...	<p>Refresh the target outline with all or some dimension changes made to the source outline.</p> <ul style="list-style-type: none"> ● add: Refresh with added dimensions. ● delete: Refresh by deleting dimensions. ● rename: Refresh with renamed dimensions. ● update: Refresh with dimensions that have member updates (required if the statement will also use apply on member). ● move: Refresh the order of dimensions in the outline. <p>Use commas to separate the types of source dimension changes to refresh on the target. For example, to refresh only with added or moved dimensions, use the following phrase: <code>apply on dimension add, move</code>.</p>
apply on member...	<p>Refresh the target outline with all or some physical member changes made to the source outline. Requires apply on dimension update.</p> <ul style="list-style-type: none"> ● add: Refresh dimensions with added members. ● delete: Refresh dimensions by deleting members. ● rename: Refresh dimensions with renamed members. ● move: Refresh the order or hierarchy of members in the dimension. <p>Use commas to separate the types of source member changes to refresh on the target. For example, to refresh only with added or moved members, use the following phrase: <code>apply on dimension update, apply on member add, move</code>.</p>
apply on member_property...	<p>Refresh the target outline with all or some member property changes made to the source outline. Requires apply on dimension update.</p> <ul style="list-style-type: none"> ● account_type: Refresh with changes in account type. ● alias: Refresh with changes to aliases. ● calc_formula: Refresh with changes to member formulas. ● consolidation: Refresh with changes to consolidation tags. ● currency_conversion: Refresh with changes to currency conversion flags. ● currency_category: Refresh with changes to currency categories. ● data_storage: Refresh with changes to data storage tags. ● uda: Refresh with changes to UDAs. <p>Use commas to separate the types of source member-property changes to refresh on the target. For example, to refresh only with updated member formulas, use the following phrase: <code>apply on dimension update, apply on member_property calc_formula</code>.</p>

Example

```
refresh outline on replicated partition sampeast.east to samppart.company
  apply all;
```

Refreshes the target outline (for Samppart.company database) with any and all changes made to the source outline (Sampeast.east).

```
refresh outline on replicated partition Sampeast.east to Samppart.company  
apply on dimension update, apply on member rename, apply on member_property  
account_type;
```

Refreshes the target outline (for Samppart.company database) with changes made to the source outline (Sampeast.east), reflecting the following update to a dimension: a member tagged Accounts was renamed.

Refresh Replicated Partition

Refresh the current replicated-partition database target from the remote (second DBS-NAME) source partition. Database Manager permission for each database is required.

Syntax

```
►► refresh replicated partition DBS-NAME [ to DBS-NAME ] [ from DBS-NAME ]  
[ at HOST-NAME ] [ all | updated ] data
```

You can update a replicated-partition database using **refresh replicated partition**.

Keyword	Description
...to...	Use the current replicated-partition database source to refresh the remote target partition.
...from...	Refresh the current replicated-partition database target from the remote source partition.
...updated data	Refresh a replicated-partition database only with data that has been updated since the last refresh.
...all data	Refresh a replicated-partition database with all data, regardless of the last refresh.

Example

```
refresh replicated partition sampeast.east to samppart.company at localhost all data;
```

MaxL Definitions

This section contains the following topics:

- [“MaxL Syntax Notes” on page 768](#)
- [“Numbers in MaxL Syntax” on page 769](#)
- [“Terminals” on page 769](#)
- [“Privileges and Roles” on page 813](#)
- [“Quoting and Special Characters Rules for MaxL Language” on page 816](#)

MaxL Syntax Notes

The following syntax scheme applies to the creation of MaxL statements.

A MaxL **statement** corresponds to a sentence telling Essbase what to do with users and database objects. In this documentation, the grammar of MaxL statements is illustrated using [railroad diagrams](#).

When issued via the MaxL Shell (essmsh), statements must be terminated by semicolons. Semicolons are used only to tell the shell when to terminate the statement; semicolons are not part of the MaxL language itself. Therefore, when issuing MaxL statements programmatically through Perl or API programs, *do not* terminate with a semicolon.

A **token** is a delimited sequence of characters recognized by MaxL as a single readable unit. Tokens may be singleton names, keywords, strings, or numbers. Names can have one, two, or three tokens, delimited by periods. The space delimiting tokens can be any white space: spaces, tabs, new lines, or blank lines.

A **keyword** is a sequence of alphabetic characters that is part of the MaxL grammar. Each keyword is recognized as one token. To be recognized as keywords, keywords cannot be enclosed in quotation marks. However, if you wish to use MaxL keywords outside of the grammar as *terminals* (for example, as database names or passwords), they must be enclosed in single or double quotation marks.

A **terminal** is something referenced in the grammar for which you provide the correct name or definition. Terminals can be names, numbers, or strings. Examples: user-name, filter-name, size-string.

A **name** is any string that starts with an alphabetic character, or any quoted string. Names in MaxL are used to uniquely identify databases and database objects, such as users, applications, or filters.

Names in MaxL may be one of three types:

- *singletons*, which are names with one token (example: `Sample`). Use a singleton name for objects that have a system-wide context: for example, applications.
- *doubles*, which are names with two tokens. A double is two names connected by a period (example: `Sample.basic`). Use doubles to name objects with application-wide contexts, such as databases.
- *triples*, which are names with three tokens. A triple is three names connected by two periods (example: `Sample.Basic.Calcname`). Use triples to name objects having database-wide contexts, such as filters.

A **string** is unquoted or quoted. An unquoted string can be any sequence of non-special characters. A quoted string can be any sequence of characters (special, alphabetic, or numeric) in the MaxL Alphabet, enclosed in single or double quotation marks.

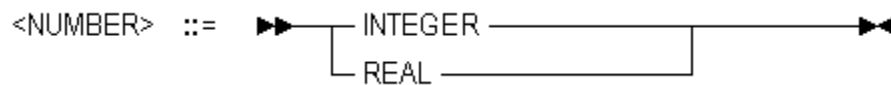
A **number** is one kind of token which may be passed to Essbase by MaxL. To have meaning, the number must be in the correct format for the Essbase value it represents. In the MaxL grammar documentation, labels for numbers indicate whether the allowed number is positive, negative, an integer, or a real. See [“Numbers in MaxL Syntax” on page 769](#).

The MaxL **alphabet** consists of the following elements:

Element	Description
Special characters	Valid special characters: . , ; : % \$ " ' SPACE TAB * + - = < > [] { } () ? ! / \ ~ ` # & @ ^ When using special characters in MaxL terminals, note the quoting rules (see “Quoting and Special Characters Rules for MaxL Language” on page 816).
Non-special characters	Alphabetic characters and numbers.
Alphabetic characters	Letters of the alphabet, and the underscore. [a-z, A-Z, _]
Numbers	See “Numbers in MaxL Syntax” on page 769

Numbers in MaxL Syntax

Numbers in MaxL statements fit into one of the following categories.



- **INTEGER**—Zero or a positive integer. Decimals and scientific notation are permitted.
Examples: 0, 1, 1000, 1.3e4
- **REAL**—Zero or a positive real number. Decimals and scientific notation are permitted.
Examples: 0.0, 1, 1000, 1000.4, 13.1e-4

Terminals

The following sections describe terminals in alphabetical order.

ACCESS-TYPE

The domains that a user can access based on the license. The only possible input value for this string is `Essbase`.

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Example

`Essbase`

Referenced By

[Alter User](#)

ACTION

The required action if a data-monitoring trigger is activated.

Syntax

```
mail [smtp], [sender], [receiver1, reciever2, ...], [subject]
spool FILE-NAME
```

- mail - sends an email from the specified sender, to a specified email address or addresses, with the specified subject line (optional). Enclose email addresses containing special characters in square brackets ([]). The mail action is not supported for after-update triggers, which are the only triggers available for use with aggregate storage databases.
- spool - logs a message in a specified file in the \$ARBORPATH\app\appname\dbname\trig folder.

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Example

```
mail manager.sales.com, [mktdir@CC.com, Monitor@acnts.com]
spool "trgmonitor"
```

Referenced By

[create trigger](#)

[drop trigger](#)

ADMIN-SVCS-LOCATION

The name (or IP address) and port number of the computer on which Essbase Administration Server runs.

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Example

```
Aspen:10080      127.0.0.1:10080
```

Referenced By

[alter system](#)

ALG-CLASS

The Java class and the method representing a data-mining algorithm. Must be the fully qualified name of the Java class that contains the logic for the algorithm. Must be enclosed in single or double quotation marks.

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Example

```
'com.hyperion.essbase.algorithms.Regression'
```

Referenced By

[create algorithm](#)

ALG-MODE

The task mode for the data-mining algorithm. Possible values for this string are: `build`.

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Referenced By

[display algorithm](#)

ALG-NAME

The name of a data mining algorithm. If the name contains special characters (see [“MaxL Syntax Notes” on page 768](#)), it must be enclosed in single or double quotation marks.

Type

name (see [“MaxL Syntax Notes” on page 768](#))

Example

```
AssocRules Regression  
'Naive Bayes'
```

Referenced By

[create algorithm](#)

[create model](#)

[display algorithm](#)

[drop algorithm](#)

ALT-NAME-SINGLE

The name of an alias table. If the name contains special characters (see [“MaxL Syntax Notes” on page 768](#)), it must be enclosed in single or double quotation marks.

Type

name (see [“MaxL Syntax Notes” on page 768](#))

Example

```
Region  
'Long Names'
```

Referenced By

[alter database](#)

[query database](#)

APP-NAME

The name of the application. Limit 8 characters.

If the name contains any allowed special characters, it must be enclosed in single or double quotation marks. Only the following special characters are allowed by Essbase within application names:

```
% (percent sign)  
$ (dollar sign)  
- (minus sign)  
{ (open brace)  
} (close brace)  
( (open parenthesis)  
) (close parenthesis)  
! (exclamation mark)  
~ (tilde)  
` (accent mark)  
# (pound sign)  
& (ampersand)  
@ (at sign)  
^ (caret)
```

Type

name (see [“MaxL Syntax Notes” on page 768](#))

Example

Sample

Referenced By

[alter application](#)

[alter partition](#)

[alter system](#)

[create application](#)

[display application](#)

[display calculation](#)

[display database](#)

[display function](#)

[display location alias](#)

[display lock](#)

[display macro](#)

[display object](#)

[display session](#)

[display trigger spool](#)

[drop application](#)

[drop lock](#)

[grant](#)

[refresh custom definitions](#)

[query application](#)

AREA-ALIAS

A shorthand name used in the in the [create partition](#) statement for referring to an already-specified member expression that designates which areas of the databases should be partitioned.

Type

name (see [“MaxL Syntax Notes”](#) on page 768)

Example

In the create partition statement below, "foo" is an area-alias for the member expression specified in the area specification. To create area-aliases, enter the alias names after the member expression in each area specification. To specify which area is relevant when mapping members (if applicable), refer to its alias name in the **mapped** phrase.

In the example below, the alias name as *created* is shown in this color, and it specifies which area (in other words, it refers to the entire member expression string, '@IDESCENDANTS (East) @IDESCENDANTS (Qtr1) '). The alias name as *referenced* is shown in this color.

```
create or replace replicated partition sampeast.east
  area '@IDESCENDANTS("Eastern Region"), @IDESCENDANTS(Qtr1) '
to samppart.company at aspen
as admin identified by 'password'
  area '@IDESCENDANTS(East) @IDESCENDANTS(Qtr1) ' foo
  mapped foo (Year) to (Yr)
update allow validate only;
```

Note: All area aliases used in a mapping should be associated with the target (as in the example above), and the direction of member names listed in the mapped clause should go from source to target.

Referenced By

[create partition](#)

BUFFER-ID

A number between 1 and 999,999 inclusive. To destroy a buffer before a data load is complete, you must use the same BUFFER-ID number that was used to initialize the buffer.

Type

number (see [“MaxL Syntax Notes” on page 768](#))

Referenced By

[alter database](#)

CALC-NAME

A stored calculation.

Syntax

name1.name2.name3 (db-level calc)

OR

name1.name3 (app-level calc)

- *name1* - Application name.
- *name2* - Database name (not required for application-level calcs).
- *name3* - Calc script name.

Type

name (see [“MaxL Syntax Notes” on page 768](#))

For calculations associated with databases, three tokens are required, to indicate application and database context and the calculation name.

Example

```
Sample.basic.'alloc.csc'
```

For application-level calculations, two tokens are required, indicating application context and the calculation name. When executing application-level calculations, you must specify which database to calculate using the syntax 'on database STRING.'

Example

- `Sample.'alloc.csc'` is the application-level CALC-NAME.
- `execute calculation Sample.'alloc.csc' on database Basic;` is a way to execute the application-level calculation on a database.

If any part of the name contains special characters (see [“MaxL Syntax Notes” on page 768](#)), it must be enclosed in single or double quotation marks.

Referenced By

[create calculation](#)

[display calculation](#)

[drop calculation](#)

[execute calculation](#)

[grant](#)

CALC-NAME-SINGLE

A stored calculation name that is the third token of a database-level [“CALC-NAME” on page 774](#).

If any part of the name contains special characters (see [“MaxL Syntax Notes” on page 768](#)), it must be enclosed in single or double quotation marks.

Type

name (see [“MaxL Syntax Notes” on page 768](#))

Example

If the full database-level calc name is `sample.basic.'alloc.csc'`, then CALC-NAME-SINGLE is `'alloc.csc'`.

Referenced By

[alter database](#)

CALC-SPEC-STRING

An optional Essbase calculator-syntax specification string. Must be enclosed in single quotation marks.

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Example

```
'@COVARIANCE (expList1, expList2)'
```

Use CALC-SPEC-STRING only if the function or macro needs to be returned through the API that lists functions.

Referenced By

[create function](#)

[create macro](#)

CALC-STRING

A calculation string. The body of an anonymous (unstored) calculation, or the string used to specify the body of a stored calculation at create time.

Because calculations are terminated with a semicolon, and semicolons are special characters to MaxL, CALC-STRING should be enclosed in single or double quotation marks.

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Example

```
CALC DIM(Year, Measures, Product);
```

Referenced By

[alter database](#)

[execute calculation](#)

COLUMN-WIDTH

A number (at least 8) representing character-width of columns; or, the keyword **default**, representing 20 characters wide.

Type

number (see [“MaxL Syntax Notes” on page 768](#)) or **default**

Example

```
set display column width 80  
set display column width default
```

Referenced By

[“Set Display Column Width” on page 835](#)

COMMENT-STRING

A string of user-defined informational text. If the string contains special characters (see [“MaxL Syntax Notes” on page 768](#)), it must be enclosed in single or double quotation marks.

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Example

```
'This is a comment.'
```

Referenced By

[alter application](#)

[alter database](#)

[alter group](#)

[alter user](#)

[create application](#)

[create database](#)

[create function](#)

[create group](#)

[create macro](#)

[create partition](#)

[create user](#)

CONDITION

A numeric-value-expression developed in MDX. Must be enclosed in double quotation marks. Enclose strings containing special characters in square brackets ([]).

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Example

```
"Jan>20"
```

Referenced By

[create trigger](#)

CUBE-AREA or MDX-SET

A cube area or other specification developed in [MDX](#) as a symmetric, syntactically-valid set. The area specification must be static, for example it cannot contain Dynamic Calc members or runtime functions such as Filter, TopSum, or BottomSum. Enclose strings containing special characters in square brackets ([]). For complete information about defining MDX sets, see [“MDX Set Specification” on page 953](#) in the MDX section.

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Examples

The following is a set of siblings.

```
'{[Jan 2000], [Feb 2000], [Mar 2000]}'
```

The following is a crossjoined set.

```
'{([Qtr1], [New York]), ([Qtr1], [California]),  
 ([Qtr2], [New York]), ([Qtr2], [California])}'
```

The following set is also a tuple.

```
'{(Jun, FY2011, Actual)}'
```

The following statement clears data from a region of ASOsamp.Sample. The region is defined using a CUBE-AREA expressed in MDX.

```
alter database ASOsamp.sample clear data in region '{(Coupon, [Prev Year], South)}'  
physical;
```

Referenced By

[create trigger](#)

[alter database \(aggregate storage\)](#)

[execute allocation \(aggregate storage\)](#)

[execute calculation \(aggregate storage\)](#)

CUBE-SCHEMA-PATH

The path to the cube schema in Essbase Studio from the root folder.

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Example

```
"\folderinpath1\folderinpath2\cubeschemaname"
```

Referenced By

[deploy](#)

DATE

A valid date string formatted according to these rules:

- MM/DD/YYYY or MM/DD/YY
- Any character can be used as a separator; for example, MM~DD~YY is valid.

If the string contains special characters (see [“MaxL Syntax Notes” on page 768](#)), it must be enclosed in single or double quotation marks.

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Example

```
'04/16/03'  
'04.16.2003'  
04_16_2003
```

Referenced By

[alter database](#)

[query database](#)

DBS-EXPORT-DIR

Suffix for the name of a database directory to contain export files, to be created (upon [export lro](#)) on the server or client as `$ARBORPATH/app/appname-dbname-suffix`.

After [export lro](#), the directory contains file-type LRO binary files (if applicable to the database), and the LRO-catalog export file with file-extension `.exp`.

If for a Sample.Basic export, DBS-EXPORT-DIR is given as `lros`, then the `sample-basic-lros` directory is created in the `$ARBORPATH/app` directory structure. The `sample-basic-`

lros directory contains file-type LRO binary files and the LRO-catalog export file 'sample-basic-lros.exp'.

Notes:

- MaxL creates exactly one export directory; it does not create a directory *structure*.
- If the specified export directory already exists, the export LRO statement will fail. This is a safeguard against overwriting existing export directories.

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Referenced By

[export lro](#)

DBS-NAME

The name of a database. Two tokens are required, to indicate application context.

Syntax

name1.name2

- *name1* - The name of the application containing the database. Limit 8 characters.
- *name2* - The name of the database. Limit 8 characters.

If the name contains any allowed special characters, it must be enclosed in single or double quotation marks. Only following special characters are allowed by Essbase within database names:

```
% (percent sign)
$ (dollar sign)
- (minus sign)
{ (open brace)
} (close brace)
( (open parenthesis)
) (close parenthesis)
! (exclamation mark)
~ (tilde)
^ (accent mark)
# (pound sign)
& (ampersand)
@ (at sign)
^ (caret)
```

Type

name (see [“MaxL Syntax Notes” on page 768](#))

Example

```
Sample.basic
```

Referenced By

alter database
alter partition
alter system
alter trigger
create database
create location alias
create mining result
create model
create outline
create partition
display database
display disk volume
display filter
display filter row
display location alias
display lock
display mining result
display mining task template
display model
display object
display partition
display session
display trigger spool
display variable
drop database
drop lock
drop partition
drop trigger spool
execute aggregate build
execute aggregate process
execute aggregate selection
export data

[grant](#)
[import data](#)
[import dimensions](#)
[import lro](#)
[query database](#)
[refresh outline](#)
[refresh replicated partition](#)

DBS-STRING

The second token of “[DBS-NAME](#)” on page 780. Limit 8 characters.

If the name contains special characters (see “[MaxL Syntax Notes](#)” on page 768), it must be enclosed in single or double quotation marks.

Type

string (see “[MaxL Syntax Notes](#)” on page 768)

Example

`basic`

Referenced By

[alter application](#)
[alter database](#)
[alter partition](#)
[execute calculation](#)

DIM-NAME

The name of a database dimension.

If the string contains special characters (see “[MaxL Syntax Notes](#)” on page 768), it must be enclosed in single or double quotation marks.

Type

string (see “[MaxL Syntax Notes](#)” on page 768)

Example

`Year`
`Market`

Referenced By

[query database](#)

ESS-CONN

The name of an Essbase connection stored on the Essbase Studio Server.

Referenced By

[deploy](#)

ESS-MODEL-NAME

The name of an Essbase model on the Essbase Studio Server.

Referenced By

[deploy](#)

EXPORT-DIR

The exact name of a directory in `$ARBORPATH\app` where LRO-catalog information was exported using [Export LRO](#). Give only the directory name; do not give the full path. Must be enclosed in single or double quotation marks. The typical format is `appname-dbname-suffix`.

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Example

```
'sample-basic-out'
```

Referenced By

[alter system](#)

FILE-NAME

A file name or an absolute path to a file. If the string contains special characters (see [“MaxL Syntax Notes” on page 768](#)), it must be enclosed in single or double quotation marks. Double quotation marks allows variable expansion; single quotation marks does not. If the file path contains a backslash (`\`), it must be preceded with another backslash (`\\`) to be interpreted correctly by the MaxL Shell.

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Example

- file01
- 'D:\\filename'
- "\$ARBORPATH/errors.txt"
- "\$ARBORPATH\\app\\sample\\basic\\calcdat.txt" (double quotation marks to expand the variable)
- '/homes/fiona/scriptfile.msh' (UNIX file path)

Referenced By

[alter database](#)

[create mining task template](#)

[create model](#)

[export data](#)

[export model](#)

[import data](#)

[import dimensions](#)

FILE-NAME-PREFIX

Prefix for one or more file names to be created (upon **display drillthrough DBS-NAME to FILE-NAME-PREFIX**) on the client in the working directory of MaxL execution.

These display output files contain the URL XML content of URL drill-through definitions used to link to content hosted on ERP and EPM applications.

If the string contains special characters (see [“MaxL Syntax Notes” on page 768](#)), it must be enclosed in single or double quotation marks.

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Example

```
urlxmls
```

Referenced By

[display drillthrough](#)

FILTER-NAME

The name of a security filter. Three tokens are required, to indicate application and database context.

Syntax

name1.name2.name3

- *name1* - Application name.
- *name2* - Database name.
- *name3* - Filter name.

Type

name (see [“MaxL Syntax Notes” on page 768](#))

Example

```
Sample.basic.filt1
```

Referenced By

[alter filter](#)

[create filter](#)

[display filter](#)

[display filter row](#)

[drop filter](#)

[grant](#)

FULL-EXPORT-DIR

Full path for the name of a directory for LRO export files, to be created (upon [export lro](#)) anywhere on the client or server.

After [export lro](#), the directory contains file-type LRO binary files (if applicable to the database), and the LRO-catalog export file named in the format *directoryname.exp*.

For example, if for a Sample.Basic export, FULL-EXPORT-DIR is given as `home/temp/lros`, then the `lros` directory structure is created under `home/temp` if `home/temp` exists. The `lros` subdirectory contains file-type LRO binary files and the LRO-catalog export file '`lros.exp`'.

Notes:

- MaxL creates exactly one export directory; it does not create a directory *structure*. In the above example, if the `home/temp` directory structure exists, MaxL creates the `lros` directory as a subdirectory of `home/temp`, but if `home/temp` does not exist, MaxL will not create `home/temp/lros`.
- If the specified export directory already exists, the export LRO statement will fail. This is a safeguard against overwriting existing export directories.
- On Windows, use double backslashes (`\\`) to represent backslashes in file paths. This is so that the MaxL Shell can interpret the second backslash literally, and not as an escape sequence.

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Example

```
'C:\\temp\\lros'
```

Referenced By

[Export LRO](#)

FUNC-NAME

The name of a custom-defined Essbase function. Using one token indicates a global function. For a local (application-level) function, use two tokens.

The name of a custom-defined function is a unique string that begins with a letter or a @, #, \$, _ symbol. The name can include alphanumeric characters or the aforementioned symbols. It is recommended that you start a function name with @.

Any token of the name that contains special characters (see [“MaxL Syntax Notes” on page 768](#)), must be enclosed in single or double quotation marks.

Syntax

name1.name2 (local)

OR

name2 (Global)

See [“MaxL Syntax Notes” on page 768](#)

- *name1* - Application name.
- *name2* - Function name.

Type

name (see [“MaxL Syntax Notes” on page 768](#))

Example

- Sample. '@COVARIANCE' (a local function)
- '@COVARIANCE' (a global function)

Referenced By

[display function](#)

[drop function](#)

GROUP-NAME

The name of the Essbase security group. If the group is authenticated with Shared Services, the name must match a valid group name on one of the configured authentication repositories.

Group name guidelines:

- Non-Unicode application limit: 256 bytes
- Unicode-mode application limit: 256 characters
- Group names must start with a letter or a number
- If the group name contains any special characters (see [“MaxL Syntax Notes” on page 768](#)), the name must be enclosed in single or double quotation marks.

When Essbase is in EPM System security mode, GROUP-NAME can include a user directory specification or unique identity attribute.

In EPM System security mode, user and group names can be non unique, if you specify either the user or group's provider directory or unique identity attribute.

Types

- name (see [“MaxL Syntax Notes” on page 768](#))
- name@provider
- WITH IDENTITY [ID-STRING](#)

where *provider* is the name of a user directory (such as LDAP or Active Directory) that hosts the external group, and [ID-STRING](#) is a unique identity assigned to every user and group (if Essbase is in EPM System security mode).

Note: If a user or group name includes the @ character, you must specify the provider as well, or else Shared Services considers the @ character as a delimiter indicating a provider name. For example, if you want to log in user admin@msad which is on a Native Directory provider, you must specify 'admin@msad@Native Directory'.

Examples

```
Sales010
```

```
Sales010@Native Directory
```

```
with identity "native://nvid=f0ed2a6d7fb07688:5a342200:1265973105c:-7f46?GROUP"
```

Referenced By

[alter application](#)

[alter group](#)

[alter user](#)

[create group](#)

create user
display group
display privilege
display user
drop group
grant

HOST-NAME

The name of a computer. The maximum length of a computer name can be 1024 bytes (non-Unicode application) or characters (Unicode application).

For Essbase failover clusters, you must use the URL-based Essbase Server name for the host name:

```
http[s]://host:port/aps/Essbase?clusterName=logicalName
```

For secure mode (SSL), the URL syntax is

```
http[s]://host:port/aps/Essbase?ClusterName=logicalName&SecureMODE=yesORno
```

For example,

```
https://myhost:13080/aps/Essbase?clustername=Essbase-Cluster1&SecureMODE=Yes
```

You can optionally use IP addresses in place of host names when creating, dropping, or altering partition definitions. For example: '127.0.0.1'.

If you are creating, altering, or dropping a partition to or from another agent on the same computer, see [“Specifying Port Numbers in Partition Host Names” on page 923](#) for more information.

If you are using host name aliases, see [“Using Host Name Aliases When Partitioning” on page 924](#).

For information about partitioning in secure mode (SSL), see also [“Partitioning and SSL” on page 925](#).

Leading or trailing spaces in the host name are illegal and will be trimmed off.

Type

name (see [“MaxL Syntax Notes” on page 768](#))

ID-RANGE

A comma-separated list of sequence ID ranges for logged sequential transactions. A range can consist of:

- A single transaction: n to n ; for example, 1 to 1
- Multiple transactions: x to y ; for example, 20 to 100

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Example

```
1 to 10,20 to 100
```

Referenced By

[alter database](#)

ID-STRING

Unique identity attribute identifying a user or group in a directory.

A unique identity attribute, or "identity," is a unique string assigned to every user and group when Essbase is in EPM System security mode. The identity enables Essbase to distinguish between users and groups with the same name across providers.

To find the identities of existing users or groups, use [display user](#) or [display group](#).

For more information about unique identity attributes, see *Oracle Hyperion Enterprise Performance Management System Security Administration Guide*.

Example

```
native://nvid=f0ed2a6d7fb07688:5a342200:1265973105c:-7f46?USER
```

Referenced By

[USER-NAME](#)

[GROUP-NAME](#)

IMPORT-DIR

A string representing the full path to the directory used in the **export lro** statement.

Note: If importing lros from a server directory (using **from server** syntax of **import lro**), you can give just the full directory name instead of the full path, as specified by [“EXPORT-DIR” on page 783](#).

The string must be enclosed in single or double quotation marks.

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Example

- 'C:\\Hyperion\\products\\Essbase\\EssbaseServer\\app\\sample-basic-lros'
- 'home/exports/temp/sample-basic-lros'
- "\$ARBORPATH\\app\\sample-basic-lros"

Note: If variables are used, the string should be enclosed in double quotation marks.

For information about how IMPORT-DIR is created, see the grammar and definitions for [export lro](#).

Referenced By

[import lro](#)

IMP-FILE

A name or absolute path to a server-side rules file or data file, used for [import data](#) and [import dimension](#) statements.

If the data or rules file is specified to be on the server, the following rules apply. If the data or rules file is specified to be local (or left unspecified, in which case it is also local), skip the following and use “[FILE-NAME](#)” on page 783.

If you are using `server data_file` or `server rules_file`, you can get the file from any application (not just the current application) by starting the IMP-FILE string using the following pattern:

```
FILE_SEP AppName FILE_SEP DbName FILE_SEP rest_of_file_name
```

where FILE_SEP must be either / or \\.

Type

name (see “[MaxL Syntax Notes](#)” on page 768)

Examples

Consider the MaxL statement

```
import database demo.basic data
from server rules_file 'IMP-FILE'
on error abort;
```

If IMP-FILE is 'calcdat.txt', the file will be looked for in \Demo\Basic\calcdat.txt.

If IMP-FILE is '/Sample/Basic/calcdat.txt' (or '\\Sample\\Basic\calcdat.txt'), the file will be looked for in \Sample\Basic\calcdat.txt.

If the *FILE_SEP string FILE_SEP string FILE_SEP* pattern does not start the string, the entire string is used as the filename, but the current application directory is assumed. For example, if the initial file separator is omitted and IMP-FILE is incorrectly specified as 'Sample/Basic/calcdat.txt', the file will be looked for in /Demo/Basic/Sample/Basic/calcdat.txt.

```
import database demo.basic data
from server file '/Sample/Basic/Calcdat.txt'
on error abort;
```

Essbase looks for calcdat.txt inside the Sample.Basic directory, and loads the data to Demo.Basic.

Referenced By

[import data](#)

[import dimensions](#)

JAVACLASS.METHOD

The java class and the method representing the custom-defined function. Must be a fully qualified java method name and signature, enclosed in single or double quotation marks.

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Example

```
'com.hyperion.essbase.calculator.Statistics.covariance'
```

For Java code examples and MaxL registration scripts for custom-defined functions, see [Custom-Defined Calculation Function Examples](#)

Referenced By

[create function](#)

LOCATION-ALIAS-NAME

The name of a location alias referencing another database.

Syntax

name1.name2.name3

- *name1* - Application name.
- *name2* - Database name.
- *name3* - Location alias name.

Type

name (see [“MaxL Syntax Notes” on page 768](#))

Example

```
Sample.Basic.EasternDB
```

Referenced By

[create location alias](#)

[display location alias](#)

LOC-ALIAS-SINGLE

The single form of a location alias name. Use if you are creating a new location alias.

Type

name (see [“MaxL Syntax Notes” on page 768](#))

Example

```
EasternDB
```

Referenced By

[alter database](#)

[create location alias](#)

LOG-TIME

A specific log time after which to replay subsequent transactions. Enclose the value in quotation marks.

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Example

```
'11_20_2007:12:20:00'
```

Referenced By

[alter database](#)

MACRO-EXPANSION

Extended definition of the macro, to be substituted in wherever the registered macro name is referenced in a calculation. If the string contains special characters (see [“MaxL Syntax Notes” on page 768](#)), it must be enclosed in single or double quotation marks.

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Example

```
'@COUNT (SKIPMISSING, @RANGE (@@S) ) '
```

For more information, see [“Custom-Defined Macros” on page 292](#).

Referenced By

[create macro](#)

MACRO-NAME

The name of a custom-defined Essbase macro. Macro names are a shorthand way to refer to macro expansions.

The name of a macro is a unique string that begins with a letter or a @, #, \$, _ symbol. The name can include alphanumeric characters or the aforementioned symbols. It is recommended that you start a macro name with @. Although macros must have unique names within a given application, a global macro and a local macro can share the same name. However, the local macro takes precedence.

To create or refer to a local (application-level) macro, use the double name (for example, **Sample. '@JSUM'**).

Any part of the name that contains special characters (see [“MaxL Syntax Notes” on page 768](#)), must be enclosed in single or double quotation marks.

Syntax

name1.name2 (local)

OR

name2 (global)

- *name1* - Application name.
- *name2* - Macro name.

Type

name (see [“MaxL Syntax Notes” on page 768](#))

Example

- **Sample. '@COUNTRANGE'** - Application-level (local) macro name without a signature, meaning that there are no restrictions on its arguments.
- **Sample. '@COUNTRANGE (Any)'** - Same as **Sample. '@COUNTRANGE'**. Once registered for the application, @COUNTRANGE can take any arguments.
- **'@JCOUNTS'** - System-level (global) macro name.
- **'@JCOUNTS (single, group)'** - Same as **'@JCOUNTS'**, but with a signature restricting its arguments.

For more information about macro signatures (input parameters), see [“Custom-Defined Macro Input Parameters” on page 292](#)

Referenced By

[create macro](#)

[display macro](#)

[drop macro](#)

ALLOC-NUMERIC

An MDX numeric value expression used to specify the amount for an allocation source. The amount value is allocated to cells in the target region. The allocation numeric is one of the following:

- An MDX tuple
- A number
- An arithmetic expression using member names, with the following restrictions:
 - All members in the expression must be from the same dimension.
 - Tuples cannot be used.
 - Only arithmetic operators (+, -, /, and *) can be used.
 - MDX functions (such as Avg and Parent) are not allowed.

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Examples

- `(Acc_1000, Jan_2009)`
- `100.00`
- `(Acc_1000 + Acc_2000)/2`
- `AcctA + AcctB`
- `Balance * 1.1`

Referenced By

[execute allocation \(aggregate storage\)](#)

MEMBER-EXPRESSION

Outline member specification of members from one or more dimensions, member combinations separated by commas, or member sets defined with functions. Must be enclosed in single or double quotation marks.

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Example

```
'@ANCESTORS(Qtr2)'
```

If MEMBER-EXPRESSION contains MEMBER-NAMES that begin with numbers or contain special characters, enclose those member names in double quotation marks, and the entire MEMBER EXPRESSION in single quotation marks. For example:

- `create or replace filter demo.basic.numfilt no_access on '"2"'`;
- `'@DESCENDANTS("Eastern Region"), @CHILDREN(Qtr1)'`

The following example shows how [create drillthrough](#) uses a member expression to define the list of drillable regions.

```
create drillthrough sample.basic.myURL from xml_file "temp.xml" on  
{ '@Ichildren("Qtr1")', '@Ichildren("Qtr2")' } level0 only;
```

Referenced By

[alter filter](#)

[create filter](#)

[create partition](#)

[create drillthrough](#)

[alter drillthrough](#)

MEMBER-NAME

The name of a database outline member.

If the name contains special characters (see [“MaxL Syntax Notes” on page 768](#)), it must be enclosed in single quotation marks.

Type

name (see [“MaxL Syntax Notes” on page 768](#))

Example

- `Jan`
- `'New York'`

If MEMBER-NAME is part of [“MEMBER-EXPRESSION” on page 794](#) and MEMBER-NAME begins with a number or contains special characters (see [“MaxL Syntax Notes” on page 768](#)), enclose MEMBER-NAME in double quotation marks and enclose MEMBER-EXPRESSION in single quotation marks.

Referenced By

[alter database](#)

[create partition](#)

[query database](#)

MODEL-ACCESSOR

The entity in a data-mining model that accesses data. The data can be input, output, or model data. The accessor name reflects the type of data it accesses.

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Example

predictor or *target*

Referenced By

[display model](#)

MODEL-MODE

The task mode for the data-mining model. Possible values for this string are: `apply` or `test`.

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Referenced By

[display model](#)

MODEL-NAME

The name of a data-mining model to apply to create data mining results. The model must exist.

Type

name (see [“MaxL Syntax Notes” on page 768](#))

Referenced By

Create Model [display model](#)

[drop model](#)

[export model](#)

[create mining result](#)

OBJ-NAME

The name of a database object. Three tokens are required, to indicate application and database context.

Syntax

name1.name2.name3

- *name1* - Application name.
- *name2* - Database name.
- *name3* - Object name.

Type

name (see [“MaxL Syntax Notes” on page 768](#))

Example

`Sample.basic.Calcdat`

Referenced By

[alter object](#)

[drop object](#)

OBJ-NAME-SINGLE

A stored database object name that is the third token of a database-level [“OBJ-NAME” on page 797](#).

If any part of the name contains special characters (see [“MaxL Syntax Notes” on page 768](#)), it must be enclosed in single or double quotation marks.

Type

name (see [“MaxL Syntax Notes” on page 768](#))

Example

If the full database object name is `sample.basic.calcdat`, then OBJ-NAME-SINGLE is `calcdat`.

Referenced By

[alter object](#)

OUTLINE-ID

The numeric identification of an aggregate storage outline associated with a view. The outline ID is returned by the [execute aggregate selection](#) statement. The [execute aggregate selection](#) statement returns a set of views, including the outline ID for the views it returns.

Type

number (see [“MaxL Syntax Notes” on page 768](#))

Example

4142187876

Referenced By

[execute aggregate selection](#)

[execute aggregate build](#)

PASSWORD

A user's password. Not applicable for externally authenticated users.

Password guidelines:

- Non-Unicode application limit: 100 bytes
- Unicode-mode application limit: 100 characters
- If the string contains special characters (see [“MaxL Syntax Notes” on page 768](#)), the password must be enclosed in single or double quotation marks
- Leading or trailing spaces are illegal and will be trimmed off

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Referenced By

[alter partition](#)

[alter user](#)

[create location alias](#)

[create outline](#)

[create partition](#)

[create user](#)

[Login](#)

PATHNAME_FILENAME

An absolute path to a file. If the string contains special characters (see [“MaxL Syntax Notes” on page 768](#)), it must be enclosed in single or double quotation marks. Double quotation marks allows variable expansion; single quotation marks does not. If the file path contains a backslash (\), it must be preceded with another backslash (\\) to be interpreted correctly by the MaxL Shell.

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Example

- 'C:\\Hyperion\\products\\Essbase\\EssbaseServer\\app\\Sample\\Basic\\listoutput.csv'
- "\$ARBORPATH/errors.txt"
- "\$ARBORPATH\\app\\sample\\basic\\calcdat.txt" (double quotation marks to expand the variable)

Referenced By

[query database](#)

PRECISION-DIGITS

An integer between 0 and 15, inclusive.

Type

number (see [“MaxL Syntax Notes” on page 768](#))

Referenced By

[alter session](#)

PROPS

Aggregate storage data load properties that determine how missing and zero values, duplicate values, and multiple values for the same cell in the data source are processed.

- `ignore_missing_values`: Ignore missing values in the data source.
- `ignore_zero_values`: Ignore zeros in the data source.
- `aggregate_use_last`: Combine duplicate cells by using the value of the cell that was loaded last into the data load buffer. When using this option, data loads are significantly slower, even if there are not any duplicate values.

Caution! The `aggregate_use_last` method has significant performance impact, and is not intended for large data loads. If your data load is larger than one million cells, consider separating the numeric data into a separate data load process (from any typed measure data). The separate data load can use `aggregate_sum` instead.

- `aggregate_sum`: (Default) Add values when the buffer contains multiple values for the same cell.

If you use multiple properties and any conflict occurs, the last property listed takes precedence.

Type

string (see “[MaxL Syntax Notes](#)” on page 768)

Referenced By

[alter database](#) (aggregate storage)

RECORD-EXPR

Used for scoring, a data mining expression of data as a list of MDX tuples.

Type

string (see “[MaxL Syntax Notes](#)” on page 768)

Example

```
{ ([New_York], [Bagel], [Profit]), (New_York], [Rye_bread], [Cost]),  
  ([Boston], [Wheat_bread], [Cost]) }
```

For more information see the *Oracle Essbase Database Administrator's Guide* chapter titled "Mining an Essbase Database."

Referenced By

[query database](#)

RESULT-ACCESSOR

The entity in a data-mining result record that accesses data. The data can be input, output, or model data. The accessor name reflects the type of data it accesses.

Type

string (see “[MaxL Syntax Notes](#)” on page 768)

Example

predictor or *target*

Referenced By

[display mining result](#)

RESULT-MODE

The task mode for the data-mining result. Possible values for this string are: `apply` or `test`.

Type

string (see “[MaxL Syntax Notes](#)” on page 768)

Example

`apply`

Referenced By

[create mining result](#)

[display mining result](#)

[drop mining result](#)

RESULT-NAME

The name of a data-mining result record.

Type

name (see “[MaxL Syntax Notes](#)” on page 768)

Referenced By

[create mining result](#)

[display mining result](#)

[drop mining result](#)

RNUM

Resource usage specification for temporary aggregate storage data load buffer.

Must be a number between .01 and 1.0 inclusive. If not specified, the default value is 1.0. Only two digits after the decimal point are significant (for example, 0.029 is interpreted as 0.02). The total resource usage of all load buffers created on a database cannot exceed 1.0 (for example, if a buffer of size 0.9 exists, you cannot create another buffer of a size greater than 0.1). Send operations internally create load buffers of size 0.2; therefore, a load buffer of the default size of 1.0 will cause send operations to fail because of insufficient load buffer resources.

Type

number (see “[MaxL Syntax Notes](#)” on page 768)

Example

0.02

Referenced By

[alter database](#) (aggregate storage)

RULE-FILE-NAME

A comma separated list of strings of rules-file names. Each rules-file name should be an 8-character object file name with no extension. The rule files must reside on the Essbase server.

Type

string (see “[MaxL Syntax Notes](#)” on page 768)

Example

'h1h1h1' , 'h1h1h2'

Referenced By

[import data](#) (aggregate storage)

SESSION-ID

The unique session ID. This ID can be used to logout a user session, or kill the current request in that session.

Type

number (see “[MaxL Syntax Notes](#)” on page 768)

Example

3310545319

Referenced By

[alter system](#)

[display session](#)

[query database](#)

[display mining session](#)

SIZE-STRING

Syntax

number units

OR

number

- *number* - Any positive number. Decimals and scientific notation are permitted. Whitespace between *number* and *units* is optional.
- *units* - One of the following: b, kb, mb, gb, tb (case-insensitive). If units are unspecified, bytes are assumed.

Type

number (see [“MaxL Syntax Notes” on page 768](#))

Examples

```
51040b
51040 b
11MB
11000kb
12.34gb
1234e-2gb
```

Referenced By

[alter application](#)

[alter database](#)

[alter tablespace](#)

SPOOL-NAME

The name of a trigger's output file, as specified in the THEN or ELSE section of the [create trigger](#) statement.

Syntax

name1.name2.name3

Type

name (see [“MaxL Syntax Notes” on page 768](#))

Example

In the following create trigger statement, the **bold** section is the spool name.

```
create or replace trigger Sample.Basic.Trigger_Jan_20
where " (Jan, Sales, [100], East, Actual) "
```

```
when Jan > 20 and is(Year.currentmember,Jan) then
spool Trigger_Jan_20
end;
```

Referenced By

[display trigger spool](#)

[drop trigger spool](#)

ST-HIER

A named hierarchy in Essbase Studio.

Referenced By

[deploy](#)

ST-LEAF

A path in an Essbase Studio hierarchy leading from a top level to level 0. All levels from top to bottom must be included. Each level name must be enclosed in single quotation marks. Level names must be separated using the following character sequence: -<

Example

```
'H_Market' -< 'REGION' -< 'STATE'
```

Referenced By

[deploy](#)

STOPPING-VAL

Optional stopping value for the [execute aggregate process](#) statement. Use this value to give the ratio of the growth size you want to allow during the materialization of an aggregate storage database, versus the pre-aggregation size of the database (Before an aggregation is materialized, the database contains only level 0 input-level data.)

Type

number (see [“MaxL Syntax Notes” on page 768](#))

Example

A stopping value of 1.5 means that during the materialization of the aggregation, the aggregate cells are allowed to occupy up to 50% of the disk space occupied by the level-0 data.

Referenced By

[execute aggregate selection](#)

[execute aggregate process](#)

TABLSP-NAME

The name of a tablespace. Tablespaces are applicable only to aggregate storage databases. For this release, possible names for tablespaces you can alter are `default` and `temp`. Other tablespace names reserved by the system are `metadata` and `log`.

Syntax

name1.name2

- *name1* - Application name.
- *name2* - Tablespace name.

Type

name (see [“MaxL Syntax Notes” on page 768](#))

Example

`temp`

Referenced By

[alter tablespace](#)

[display tablespace](#)

TASK-MODE

The mode for the data-mining task template. Possible values for this string are: `build`, `apply`, `test` or `score`.

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Example

`apply`

Referenced By

[display mining task template](#)

[drop mining task template](#)

TASK-NAME

The name of a data-mining build, apply, or test template. You use a template to create the corresponding model.

Type

name (see [“MaxL Syntax Notes” on page 768](#))

Referenced By

[create mining task template](#)

[display mining task template](#)

[drop mining task template](#)

TASK-XML-STRING

A string of XML representing a data-mining task to execute. Must be enclosed in single quotation marks.

There are several ways of obtaining the XML string. For the build task, the XML can be obtained either from `display algorithm template` (for a new build template) or from `display mining template` with mode 'build' (for an existing build template). For the apply task, the XML can be obtained either from `display model template` (for a new apply template) or from `display mining template` with mode 'apply' (for an existing apply template).

You may need to do the following before pasting the XML string into a MaxL statement:

1. modify expressions in the template
2. escape all single quotation marks in the XML (replace all ' with \ ')

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Referenced By

[create mining result](#)

[create mining task template](#)

[create model](#)

TRANSF-CLASS

The Java class and the method representing a data-mining transformation. Must be the fully qualified name of the Java class that contains the logic for the transformation. Must be enclosed in single quotation marks.

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Example

```
com.hyperion.essbase.transformations.Log
```

Referenced By

[create transformation](#)

TRANSF-NAME

The name of a data-mining transformation. Must be enclosed in single quotation marks.

Type

name (see [“MaxL Syntax Notes” on page 768](#))

Referenced By

[create transformation](#)

[display transformation](#)

[drop transformation](#)

TRIGGER-NAME

The name of the trigger device created to track and respond to database updates. Trigger names must be triple names, specifying application name, database name, and trigger name (if you rename the application or database, the trigger is invalidated). Trigger names are case-insensitive, are a maximum of 30 bytes, and cannot contain special characters.

Syntax

name1.name2.name3

- *name1* - Application name.
- *name2* - Database name.
- *name3* - The name of the trigger.

Type

name (see [“MaxL Syntax Notes” on page 768](#))

Example

```
Sample.Basic.MyTrigger
```

Referenced By

[alter trigger](#)

[create trigger](#)

[display trigger](#)

[drop trigger](#)

UNIQUE-VOL-NAME

The unique name of the disk volume definition. Unlike the name used when the disk volume definition was created (“[VOLUME-NAME](#)” on page 812), the unique disk-volume name must be a triple. The first two parts of the name specify application and database context. The third part of the name, on Windows, is a drive letter. On UNIX, it is a path to the `EssbaseServer` directory.

If any part of the name contains special characters (see “[MaxL Syntax Notes](#)” on page 768), that part must be enclosed in single or double quotation marks.

If a Windows file path is used which contains a backslash (\), it must be preceded with another backslash (\\) to be interpreted correctly by the MaxL Shell. If variables are used, enclose the single-quoted string in double quotes so that the MaxL Shell knows to expand the variables.

Syntax

name1.name2.name3

- *name1* - Application name.
- *name2* - Database name.
- *name3* - Disk volume name.

Type

name (see “[MaxL Syntax Notes](#)” on page 768)

Example

```
sample.basic.'vol3/hyperion/products/Essbase/EssbaseServer'  
sample.basic.c  
sample.basic.'"$ARBORPATH\diskvol_area'"
```

Referenced By

[display disk volume](#)

URL-NAME

The name of a drill-through URL definition used to link to content hosted on Oracle ERP and EPM applications.

Syntax

name1.name2.name3

- *name1*—Application name
- *name2*—Database name
- *name3*—URL name

Type

name (see [“MaxL Syntax Notes” on page 768](#))

Example

```
Sample.basic.MyURL
```

If any part of the name contains special characters (see [“MaxL Syntax Notes” on page 768](#)), the name must be enclosed in single or double quotation marks.

Referenced By

[create drillthrough](#)

[alter drillthrough](#)

[display drillthrough](#)

[drop drillthrough](#)

USER-NAME

The name of the user. If the user is authenticated with Shared Services, the name must match a valid login name on one of the configured authentication repositories.

User name guidelines:

- Non-Unicode application limit: 256 bytes
- Unicode-mode application limit: 256 characters
- If the user name contains any special characters (see [“MaxL Syntax Notes” on page 768](#)), the name must be enclosed in single or double quotation marks.

When Essbase is in EPM System security mode, USER-NAME can include a user directory specification or unique identity attribute.

In EPM System security mode, user and group names can be non unique, if you specify either the user or group's provider directory or unique identity attribute.

Types

- name (see [“MaxL Syntax Notes” on page 768](#))
- name@provider
- WITH IDENTITY [ID-STRING](#)

where *provider* is the name of a user directory (such as LDAP or Active Directory) that hosts the external user, and [ID-STRING](#) is a unique identity assigned to every user and group (if Essbase is in Oracle Hyperion Enterprise Performance Management System security mode).

Note: If a user or group name includes the @ character, you must specify the provider as well, or else Shared Services considers the @ character as a delimiter indicating a provider name. For example, if you want to log in user admin@msad which is on a Native Directory provider, you must specify 'admin@msad@Native Directory'.

Examples

```
JWSmith
```

```
JWSmith@Native Directory
```

```
with identity "native://nvid=f0ed2a6d7fb07688:5a342200:1265973105c:-7f46?USER"
```

Referenced By

[alter application](#)

[alter database](#)

[alter partition](#)

[alter system](#)

[alter user](#)

[create location alias](#)

[create outline](#)

[create partition](#)

[create user](#)

[display privilege](#)

[display user](#)

[drop lock](#)

[drop user](#)

[grant](#)

[query database](#)

[Login](#)

VARIABLE-NAME

The name of the substitution variable. The name can only contain alphanumeric characters and the underscore (a-z A-Z 0-9 _).

Type

name (see [“MaxL Syntax Notes”](#) on page 768)

Example

curmonth

Referenced By

[alter application](#)

[alter database](#)

[alter system](#)

[display variable](#)

VIEW-FILE-NAME

An aggregation script containing information derived during aggregate view selection.

The file is created under *ARBORPATH*\app\app_name\db_name\ with a .csc extension.

Aggregation scripts are valid as long as the dimension level structure in the outline has not changed.

Executing an aggregation script (using [execute aggregate build](#)) materializes the aggregate views specified within it.

The .csc extension is optional when executing the script.

The file name can be a maximum of 8 characters in length (excluding the extension) and must not contain any of the following characters, or whitespace: : ; . , = + * ? [] | < > " ' \ /

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Referenced By

[execute aggregate selection](#)

[execute aggregate build](#)

[query database](#)

VIEW-ID

The numeric identification of an aggregate view, returned by the [execute aggregate selection](#) statement. The concept of views applies only to aggregate storage databases.

VIEW-IDs persist only as long as their associated [“OUTLINE-ID” on page 798s](#). OUTLINE-IDs change when changes are made to the outline.

Type

number (see [“MaxL Syntax Notes” on page 768](#))

Example

8941

Referenced By

[execute aggregate selection](#)

[execute aggregate build](#)

VIEW-SIZE

Approximate view size as a fraction of input data size. For example, a view size of 0.5 means that the view is 2X smaller than the input-level view. The concept of views applies only to aggregate storage databases.

Type

number (see [“MaxL Syntax Notes” on page 768](#))

Referenced By

[execute aggregate build](#)

VOL-REPL

A disk-volume replacement specification when restoring from an archive file.

Valid values are a comma-separated list of volumes to replace:

- 'VOL1' with 'VOL2'
- 'VOL3' with 'VOL4'
- 'VOL5' with 'VOL6'

Type

string (see [“MaxL Syntax Notes” on page 768](#))

Example

'C' with 'F', 'D' with 'G', 'E' with 'H'

Referenced By

[alter database](#)

VOLUME-NAME

The name of the disk volume. On Windows, a drive letter. On UNIX, a path to the `EssbaseServer` directory.

If the name contains special characters (see “[MaxL Syntax Notes](#)” on page 768), it must be enclosed in single or double quotation marks.

If a Windows file path is used which contains a backslash (\), it must be preceded with another backslash (\\) to be interpreted correctly by the MaxL Shell. If variables are used, enclose the single-quoted string in double quotes so that the MaxL Shell knows to expand the variables.

Type

name (see “[MaxL Syntax Notes](#)” on page 768)

Examples

```
'vol3/hyperion/products/Essbase/EssbaseServer'
```

```
" '$ARBORPATH\diskvol_area' "
```

Referenced By

[alter database](#)

Privileges and Roles

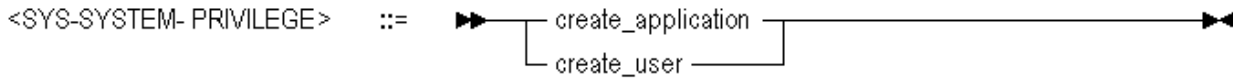
Essbase system privileges are indivisible database access types. In MaxL, privileges are grouped together to form permission-sets called *roles*. With the exception of `create_user` and `create_application`, privileges themselves are not grantable using MaxL; you typically grant roles, which are the equivalent of privilege levels. The scope of a role can be the system, the application, or the database.

While one privilege does not imply another, roles are hierarchical. The following table illustrates the Essbase system privileges that are contained in each MaxL system role.

Privileges and Roles	read	write	calculate	manage database	create database	start application	manage application	create/drop application	create/drop user
no access
read	■
write	■	■
execute	■	■	■
manager (database)	■	■	■	■
manager (application)	■	■	■	■	■	■	■	.	.
administrator	■	■	■	■	■	■	■	■	■

System-Level System Privileges

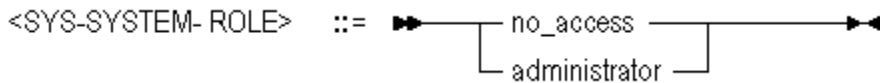
The following privileges apply at the system level. These privileges are built-in; they do not apply to any specific application or database. They are not included in any role except for the role of administrator.



- create_application—Ability to create and delete applications.
- create_user—Ability to create and delete users and groups.

System-Level System Roles

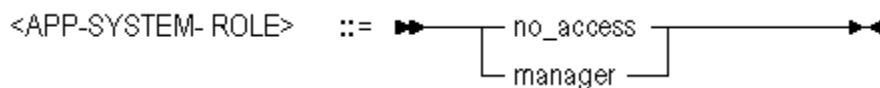
System-level system roles are applicable to the Essbase system. The following roles have a system-wide scope:



- no_access—No access to the system.
- administrator—Full access to the entire system, including other administrators.

Application-Level System Roles

Application-level system roles are applicable to an application. The following roles may have an application-wide scope:

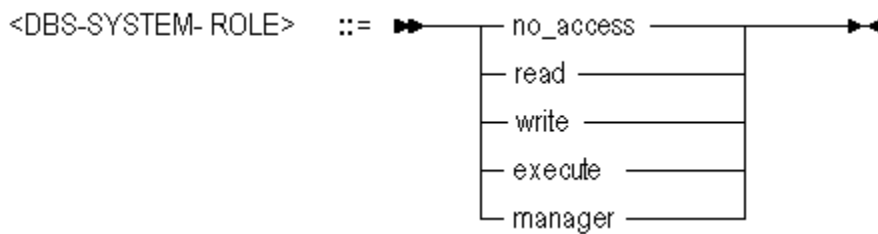


- no_access—No access to the application or any databases within it.
- manager—Manager access to the application and any databases within it. Manager access means ability to create, delete, and modify databases within the application, in addition to having Read, Write, and Execute access for that application.

Database-Level System Roles

Minimum Database Permissions

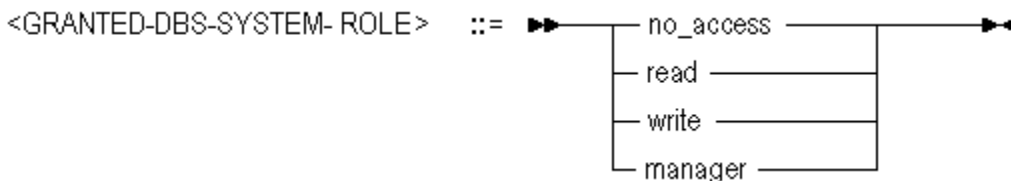
Database-level system roles are applicable to databases. The following roles have a database-wide scope and are available when assigning minimum database permissions:



- no_access—No access to the database (if assigned using [alter database](#)) or to any databases in the application (if assigned using [alter application](#)).
- read—Read-only access to the database (if assigned using [alter database](#)) or to all databases in the application (if assigned using [alter application](#)). Read access means ability to view files, retrieve data values, and run report scripts.
- write—Write access to the database (if assigned using [alter database](#)) or to all databases in the application (if assigned using [alter application](#)). Write access means ability to update data values, in addition to having Read access.
- execute—Calculate access to the database (if assigned using [alter database](#)) or to all databases in the application (if assigned using [alter application](#)). Calculate access means ability to update data values, in addition to having Read and Write access.
- manager—Manager access to the database (if assigned using [alter database](#)) or to all databases in the application (if assigned using [alter application](#)). Manager access means ability to modify database outlines, in addition to having Read and Write access.

Database Roles Grantable to Users and Groups

The following database-level system roles are available for granting to users and groups:



- no_access—No access to the database.
- read—Read-only access to the database. Read access means ability to view files, retrieve data values, and run report scripts.
- write—Write access to the database. Write access means ability to update data values, in addition to having Read access.
- manager—Manager access to the database. Manager access means ability to modify database outlines, in addition to having Read and Write access.

Note: After granting read, write, or manager privilege to a user or group, these can be revoked by subsequently granting no_access. However, to prevent users from being able to load the application, you should also grant no_access at the application level. For example:

```

/* Grant read permission on a database */
grant read on database Sample.Basic to user1;

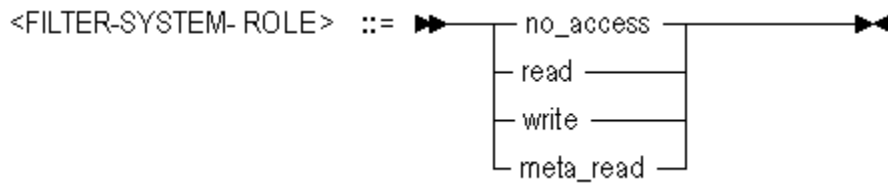
/* Revoke read permission on the database */
grant no_access on database Sample.Basic to user1;

/* Revoke read permission at the application level, to remove application-startup
permission */
grant no_access on application Sample to user1;

```

Filter Roles

The following subset of database-level system roles may be granted or revoked using filters.



- `no_access`—No access to the database.
- `read`—Read-only access to the database. Read access means ability to view files, retrieve data values, and run report scripts.
- `write`—Write access to the database. Write access means ability to update data values, in addition to having Read access.
- `meta_read`—Restricted access to sibling and ancestral metadata (dimensions and members). In case of a filtering conflict, the MetaRead filtering overrides the other filter permissions. For more information about metadata filtering, see [“Metadata Filtering” on page 926](#).

Note: After granting permissions using a filter, the permission can be revoked by subsequently granting `no_access` to the database. However, to prevent users from being able to load the application, you should also grant `no_access` at the application level. For example:

```

/* Grant read permission on a database, using a filter */
grant filter Sample.basic.filter8 to user1;

/* Revoke the filter, removing read permission on the database */
grant no_access on database Sample.Basic to user1;

/* Revoke read permission at the application level, to remove application-startup
permission */
grant no_access on application Sample to user1;

```

Quoting and Special Characters Rules for MaxL Language

These rules apply to terminals of MaxL statements; for example, `USER-NAME` or `FILE-NAME`. Rules for MaxL Shell also apply (see [“MaxL Shell Syntax Rules and Variables” on page 828](#)).

Tokens enclosed in Single Quotation Marks

Contents are preserved as literal, with the following exceptions:

- One backslash is ignored; two are treated as one.
- Apostrophe must be escaped using one backslash (\').

Example: `export database sample.basic data to data_file 'D:\\export.txt';`

Result: Exports data to D:\export.txt.

Example: `create user 'O'Brian' identified by 'password';`

Result: Error.

Example: `create user 'O\'Brian' identified by 'password';`

Result: User O'Brian is created.

Tokens Enclosed in Double Quotation Marks

Contents are preserved as literal, with the following exceptions:

- Variables are expanded.
- One backslash is ignored; two are treated as one.
- Apostrophe must be escaped using one backslash (\').

Example: `export database sample.basic data to data_file "D:\\export.txt";`

Result: Exports data to D:\export.txt.

Example: `export database sample.basic data to data_file "$ARBORPATH\\App\\Sample\\Basic\\export.txt";`

Result: Exports data to C:\Hyperion\products\Essbase\EssbaseServer\App\Sample\Basic\export.txt.

Example: `create user "O'Brian" identified by 'password';`

Result: Error.

Example: `create user "O\'Brian" identified by 'password';`

Result: User O'Brian is created.

Use of Backslashes in MaxL

Ignored unless preceded by another backslash (the escape character). Must use single or double quotation marks around the token containing the two backslashes.

`create application 'finance\\budget';`

Result: Application finance\budget is created.

Example (Windows):

```
export database sample.basic using report_file
'EssbaseServer\App\Sample\Basic\asym.rep'
to data_file 'c:\home\month2.rpt';
```

Result: The Windows file paths are interpreted correctly as EssbaseServer\App\Sample\Basic\asym.rep and c:\home\month2.rpt.

Use of Apostrophes (Single Quotation Marks)

Syntax error returned, unless preceded by a backslash (the escape character) and enclosed in single or double quotation marks.

Example: `create user 'O\'Brian' identified by 'password';`

Result: User O'Brian is created.

Note: Use sparingly. Apostrophes are permitted by Essbase in user and group names, but not in application or database names.

Use of Dollar Signs

Syntax error returned, unless preceded by a backslash (the escape character) and enclosed in single quotation marks. Dollar signs (\$) intended literally need to be escaped by the backslash so that they are not considered variable indicators.

Example: `create application '\$App1';`

Result: Application \$App1 is created.

MaxL Shell Commands

The MaxL Shell (`essmsh`) is a pre-parser mechanism for entering MaxL statements. The MaxL Shell has a separate set of useful commands, independent of the MaxL language itself. Before using any of the following MaxL Shell commands, you need to log in (see [“Login” on page 826](#)).

- [“Spool on/off” on page 833](#)
- [“Set Display Column Width” on page 835](#)
- [“Set Message Level” on page 835](#)
- [“Set Timestamp” on page 836](#)
- [“Echo” on page 836](#)
- [“Shell Escape” on page 837](#)
- [“Nesting” on page 837](#)
- [“Error Checking and Branching” on page 837](#)
- [“Version” on page 840](#)
- [“Logout” on page 840](#)

- [“Exit” on page 840](#)

Overview of MaxL Shell

The MaxL Shell (`essmsh`) is one way to execute MaxL statements or scripts. The other interfaces available for passing MaxL statements to the Essbase Server are:

- The MaxL Script Editor in the Administration Services Console. See the *Oracle Essbase Administration Services Online Help* for information about using the script editor.
- Perl programs with embedded MaxL DDL statements, made possible by adding the [“MaxL Perl Module” on page 841](#) to your Perl package.

This section contains the following topics:

- [Invocation and Login](#)
- [Syntax Rules and Variables](#)
- [Shell Commands](#)
- [“MaxL Shell and Unicode” on page 833](#)

MaxL Shell Invocation

The MaxL Shell (`essmsh`) is a pre-parser mechanism for entering MaxL statements.

Start the shell to be used interactively, to read input from a file, or to read stream-oriented input (standard input from another process). You can log in after you start the shell, interactively or using a login statement in the input file. You can also log in at invocation time, by using the `-l` flag (see [“-l Flag: Login” on page 823](#)).

- [“Prerequisites for Using MaxL” on page 819](#)
- [“MaxL Invocation Summary” on page 820](#)
- [“Interactive Input” on page 822](#)
- [“File Input” on page 825](#)
- [“Standard Input” on page 826](#)
- [“Login” on page 826](#)
- [“LoginAs” on page 827](#)
- [“Encryption” on page 827](#)
- [“Query Cancellation” on page 827](#)

Prerequisites for Using MaxL

Before the Essbase Server can receive MaxL statements,

1. The Essbase Server must be running.

2. The MaxL Shell (`essmsh`) must be invoked (see “[MaxL Invocation Summary](#)” on page 820), if you are using the shell.
3. You must log in (see “[Login](#)” on page 826) to the Essbase Server from the MaxL Shell. If you are running a MaxL script, the first line of your script must be a login statement.

When using the MaxL Shell or the MaxL Script Editor, you must use a semicolon (;) to terminate each MaxL statement.

MaxL Invocation Summary

The following MaxL Shell help page summarizes invocation options. This help is also available at the operating-system command prompt if you type `essmsh -h | more`.

```
essmsh(1)
```

NAME

```
essmsh -- MaxL Shell
```

SYNOPSIS

```
essmsh [-h|lsmup] [-a | -i | file] [arguments...]
```

DESCRIPTION

This document describes ways to invoke the MaxL Shell. The shell, invoked and nicknamed `essmsh`, takes input in the following ways: interactively (from the keyboard), standard input (piped from another program), or file input (taken from file specified on the command line). The MaxL Shell also accepts any number of command-line arguments, which can be used to represent any name.

OPTIONS

`essmsh` accepts the following options on the command line:

`-h`

Prints this help.

`-l <user> <pwd>`

Logs in a user name and password to the local Essbase Server instance.

`-u <user>`

Specifies a user to be logged in to an Essbase Server instance. If omitted but the `'-p'` or `'-s'` flags are used, `essmsh` will prompt for the username.

`-p <pwd>`

Specifies a password of the user set by the `'-u'` option to be logged in to an Essbase Server instance. If omitted, `essmsh` will prompt for the password, and the password will be hidden on the screen.

`-s <server>`

Used after `-l`, or with `[-u -p]`, logs the specified user into a named server. When omitted, `localhost` is implied.

`-m <msglevel>`

Sets the level of messages returned by the shell. Values for `<msglevel>`

are: all (the default), warning, error, and fatal.

-i

Starts a MaxL session which reads from <STDIN>, piped in from another program. The end of the session is signalled by the EOF character in that program.

-a

Allows a string of command-line arguments to be referenced from within the subsequent INTERACTIVE session. These arguments can be referenced with positional parameters, such as \$1, \$2, \$3, etc. Note: omit the -a when using arguments with a file-input session.

NOTES

No option is required to pass a filename to `essmsh`.

Arguments passed to `essmsh` can represent anything: for example, a user name, an application name, or a filter name. Arguments must appear at the end of the invocation line, following '-a', '-i', or filename.

EXAMPLES

Interactive session, simplest case:
`essmsh`

Interactive session, logging in a user:
`essmsh -l user pwd`

Interactive session, logging user in to a server:
`essmsh -l user pwd -s server`

Interactive session, logging in with two command-line arguments (referenced thereafter at the keyboard as \$1 and \$2):
`essmsh -l user pwd -a argument1 argument2`

Interactive session, with setting the message level:
`essmsh -m error`

Interactive session, hiding the password:
`essmsh -u user1`
Enter Password > *****

File-input session, simplest case:
`essmsh filename`

File-input session, with three command-line arguments (referenced anonymously in the file as \$1, \$2, and \$3):
`essmsh filename argument1 argument2 argument3`

Session reading from <STDIN>, logging into a server with two command-line arguments:
`essmsh -l user pwd -s server -i argument1 argument2`

Interactive Input

You can log into the MaxL Shell for interactive use (typing statements at the keyboard) in the following ways. See [“MaxL Invocation Summary” on page 820](#) for more descriptions of login flags.

[“No Flag” on page 822](#)

[“-a Flag: Arguments” on page 822](#)

[“-l Flag: Login” on page 823](#)

[“-u, -p, and -s Flags: Login Prompts and Hostname Selection” on page 824](#)

[“-m Flag: Message Level” on page 824](#)

No Flag

Invoked without a flag, file name, or arguments, the MaxL Shell starts in interactive mode and waits for you to log in. Note to Windows users: This is the same as double-clicking `essmsh.exe`, located in the `ESSBASEPATH\BIN` directory.

Example:

```
essmsh
```

```
Essbase MaxL Shell - Release 11.1.2
Copyright (c) 2000, 2010, Oracle and/or its affiliates.
All rights reserved.
MAXL> login Fiona identified by sunflower;

      49 - User logged in: [Fiona].
```

-a Flag: Arguments

With the `-a` flag, the MaxL Shell starts in interactive mode and accepts space-separated arguments to be referenced at the keyboard with positional parameters.

Note: If interactive arguments are used with spooling turned on, variables are recorded in the log file just as you typed them (for example, `$1`, `$2`, `$ARBORPATH`).

Example:

```
essmsh -a Fiona sunflower appname dbsname
```

```
Essbase MaxL Shell - Release 11.1.1
Copyright (c) 2000, 2008, Oracle and/or its affiliates.
All rights reserved.
```

```
MAXL> spool on to 'D:\output\createapp.out';

MAXL> login $1 identified by $2;

49 - User logged in: [Fiona].

MAXL> create application $3;

30 - Application created: ['appname'].

MAXL> create database $3.$4 as Sample.Basic;

36 - Database created: ['appname'.'dbname'].

MAXL> echo $ARBORPATH;

C:\Hyperion\products\Essbase\EssbaseClient

MAXL> spool off;
```

Contents of logfile createapp.out:

```
MAXL> login $1 identified by $2;

OK/INFO - 1051034 - Logging in user Fiona.
OK/INFO - 1051035 - Last login on Friday, January 18, 2008 4:09:16 PM.
OK/INFO - 1241001 - Logged in to Essbase.

MAXL> create application $3;

OK/INFO - 1051061 - Application appname loaded - connection established.
OK/INFO - 1054027 - Application [appname] started with process id [404].
OK/INFO - 1056010 - Application appname created.

MAXL> create database $3.$4 as Sample.Basic;

OK/INFO - 1056020 - Database appname.dbname created.

MAXL> echo $ARBORPATH;

C:\Hyperion\products\Essbase\EssbaseClient

MAXL> spool off;
```

-l Flag: Login

When the `-l` flag is used followed by a user name and password, the MaxL Shell logs in the given user name and password and starts in interactive or non-interactive mode. The user name and password must immediately follow the `-l`, and be separated from it by a space.

Example:

```
essmsh -l Fiona sunflower
```

Entered at the command prompt, this starts the MaxL Shell in interactive mode and logs in user **Fiona**, who can henceforth issue MaxL statements at the keyboard.

-u, -p, and -s Flags: Login Prompts and Hostname Selection

The MaxL Shell can be invoked using `-u` and `-p` options in interactive mode, for passing the user name and password to the shell upon startup. To be prompted for both username and password, use the `-s` option with the host name of the Essbase Server.

-s Flag: Host Name

If `-s <host-name>` is passed to the shell, MaxL will prompt for the user name and password, and the password will be hidden.

Example:

```
essmsh -s localhost
Enter UserName> admin
Enter Password> *****
```

```
OK/INFO - 1051034 - Logging in user admin.
OK/INFO - 1051035 - Last login on Monday, January 28, 2003 10:06:16 AM.
OK/INFO - 1241001 - Logged in to Essbase.
```

-u Flag: User Name

If `-u <username>` is passed to the shell and `-p <password>` is omitted, MaxL Shell will prompt for the password, and the password will be hidden.

Example:

```
essmsh -u user1
Enter Password > *****
```

-p Flag: Password

If `-p <password>` is passed to the shell and `-u <username>` is omitted, MaxL Shell will prompt for the user name.

Example:

```
essmsh -p passwrld
Enter Username > user1
```

-m Flag: Message Level

If `-m <messageLevel>` is passed to the shell, only the specified level of messages will be returned by the shell.

Example: `essmsh -m error`

Values for the `<messageLevel>` include: **default**, **all**, **warning**, **error**, and **fatal**. The default value is **all** (same as specifying **default**).

File Input

You invoke the MaxL Shell to run scripts (instead of typing statements at the keyboard) in the following ways. See “[MaxL Invocation Summary](#)” on page 820 for a complete description of login flags.

“[File Only](#)” on page 825

“[File Only](#)” on page 825

File Only

If you type `essmsh` followed by a file name or path, the shell takes input from the specified file.

Examples:

```
essmsh C:\Hyperion\products\Essbase\EssbaseClient\scripts\filename.msh
```

Entered at the command prompt, this starts the shell, tells it to read MaxL statements from a file, and terminates the session when it is finished.

```
essmsh filename
```

Starts the shell to read MaxL statements from `filename`, located in the current directory (the directory from which the MaxL Shell was invoked).

File with Arguments

If you type `essmsh` followed by a file name followed by an argument or list of space-separated arguments, `essmsh` remembers the command-line arguments, which can be referenced as `$1`, `$2`, etc. in the specified file. If spooling is turned on, all variables are expanded in the log file.

Example:

```
D:\Scripts>essmsh filename.msh Fiona sunflower localhost newuser
```

Starts the shell to read MaxL statements from `filename.msh`, located in the current directory.

Contents of script `filename.msh`:

```
spool on to $HOME\output\filename.out;
login $1 $2 on $3;
create user $4 identified by $2;
echo "Essbase is installed in $ESSBASEPATH";
spool off;
exit;
```

Contents of logfile `filename.out`:

```
MAXL> login Fiona sunflower on localhost;

      49 - User logged in: [Fiona].

MAXL> create user newuser identified by sunflower;
```

```
20 - User created: ['newuser'].
```

Essbase is installed in C:\Hyperion\products\Essbase\EssbaseClient

Standard Input

With the `-i` flag, `essmsh` uses standard input, which could be input from another process. For example,

```
program.sh | essmsh -i
```

When `program.sh` generates MaxL statements as output, you can pipe `program.sh` to `essmsh -i` to use the standard output of `program.sh` as standard input for `essmsh`. `essmsh` receives input as `program.sh` generates output, allowing for efficient co-execution of scripts.

Example:

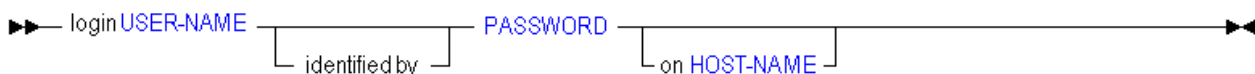
```
echo login Fiona sunflower on localhost; display privilege user;|essmsh -i
```

The MaxL Shell takes input from the `echo` command's output. User Fiona is logged in, and user privileges are displayed.

Login

Before you can send MaxL statements from the MaxL Shell to Essbase Server, you must log in to an Essbase Server session.

Note: Before logging in to an Essbase Server session, you must start the MaxL Shell (see “[MaxL Invocation Summary](#)” on page 820). Or, you can start the MaxL Shell and log in (see “[-l Flag: Login](#)” on page 823) at the same time.



Note: Login is part of the MaxL Shell grammar, not the MaxL language itself. You can use a login statement in MaxL scripts and the MaxL Shell, but you cannot embed it in Perl.

Example

```
login admin mypassword on localhost;
```

Establishes a connection to the Essbase Server for user Admin identified by mypassword.

```
login admin password on http://myhost:13080:aps/Essbase?clustername=EssbaseCluster1
```

Establishes a connection to an Essbase failover cluster for user Admin identified by password.

LoginAs

To facilitate creating scheduled reports with user-appropriate permissions, administrators can log in as another user from MaxL.

Example of "log in as" statement:

```
loginas USER-NAME PASSWORD MIMICKED-USER-NAME [on HOST-NAME];
```

Example of "log in as" invocation method:

```
essmsh -la USER-NAME PASSWORD MIMICKED-USER-NAME [-s HOST-NAME]
```

Interactive example:

```
MAXL>loginas;  
Enter UserName> username  
Enter Password> password  
Enter Host> machine_name  
Enter UserName to Login As> mimicked_user_name
```

Encryption

You can encrypt user and password information stored in MaxL scripts.

The following MaxL Shell invocation generates a public-private key pair that you can use to encrypt a MaxL script.

```
essmsh -gk
```

The following MaxL Shell invocation encrypts the input MaxL script, obscuring user name and password, and changing the file extension to .mxls.

```
essmsh -E scriptname.xml PUBLIC-KEY
```

Nested scripts are also encrypted. To avoid this and encrypt only the base script, use -Em.

The following MaxL Shell invocation decrypts and executes the MaxL script.

```
essmsh -D scriptname.mxls PRIVATE-KEY
```

The following invocation encrypts input data and returns it in encrypted form. This is useful if there is a need to manually prepare secure scripts.

```
essmsh -ep DATA PUBLIC-KEY
```

The following invocation enables you to encrypt the base script while saving any nested scripts for manual encryption.

```
essmsh -Em scriptname.xml PUBLIC-KEY
```

Query Cancellation

You can use the Esc key to cancel a query running from MaxL Shell.

MaxL Shell Syntax Rules and Variables

The MaxL Shell (essmsh) is a pre-parser mechanism for entering MaxL statements. The following syntax information can help you use the MaxL Shell successfully.

[“Semicolons” on page 828](#)

[“Variables” on page 828](#)

[“Quoting and Special Characters Rules for MaxL Language” on page 816](#)

Semicolons

When a MaxL statement is passed to Essbase Server interactively or in batch mode via the MaxL Shell (essmsh), it must be terminated by a semicolon. Semicolons are used only to tell essmsh when to terminate the statement; semicolons are not part of the MaxL language itself. Therefore, when issuing MaxL statements programmatically through Perl or API programs, do *not* use semicolons.

Examples

Program	Example
Interactive MaxL Shell	<code>create application Sample;</code>
MaxL Shell script:	<code>login \$1 identified by \$2; create application Sample; create currency database Sample.Internatl; display database Sample.Internatl; exit;</code>
Perl function (Correct)	<code>print \$dbh->do("create currency database Sample.Internatl");</code>
Perl function (Incorrect)	<code>print \$dbh->do("create currency database Sample.Internatl;");</code>

Variables

[“Overview of MaxL Shell” on page 819](#)

[“Environment Variables” on page 829](#)

[“Positional Parameters” on page 829](#)

[“Locally Defined Shell Variables” on page 830](#)

[“Quotation Marks and Variable Expansion” on page 830](#)

[“Exit Status Variable” on page 831](#)

Overview of MaxL Shell Variables

In the MaxL Shell, you can use **variables** as placeholders for any data that is subject to change or that you refer to often; for example, the name of a computer, user names, and passwords.

You can use variables in MaxL scripts as well as during interactive use of the shell. Using variables in MaxL scripts eliminates the need to create many customized scripts for each user, database, or host.

Variables can be environment variables (for example, `$ESSBASEPATH`, which references the directory Essbase is installed to), positional parameters (for example, `$1`, `$2`, etc.), or locally defined shell variables.

All variables must begin with a `$` (dollar sign). Locally defined shell variables should be *set* without the dollar sign, but should be *referenced* with the dollar sign. Example:

```
set A = val_1;
echo $A;
val_1
```

Note: Variables can be in parentheses. Example: if `$1 = arg1`, then `$(1)23 = arg123`.

Use double quotation marks around a string when you want the string interpreted as a single token with the variables recognized and expanded. For example, `"$ESSBASEPATH"` is interpreted as `C:\Hyperion\products\Essbase\EssbaseServer`.

Use single quotation marks around a string to tell `essmsh` to recognize the string as a single token, *without* expanding variables. For example, `'$ESSBASEPATH'` is interpreted as `$ESSBASEPATH`, not `C:\Hyperion\products\Essbase\EssbaseServer`.

Environment Variables

You can reference any environment variable in the MaxL Shell.

Example (Windows): `spool on to "$ESSBASEPATH\out.txt";`

Result: MaxL Shell session is recorded to `C:\Hyperion\products\Essbase\EssbaseServer\out.txt`.

Example (UNIX): `spool on to "$HOME/output.txt";`

Result: MaxL Shell session is recorded to `output.txt` in the directory referenced by the `$HOME` environment variable.

Positional Parameters

Positional parameter variables are passed in to the shell at [invocation](#) time as arguments, and can be referred to generically by the subsequent script or interactive MaxL Shell session using `$n`, where `n` is the number representing the order in which the argument was passed on the command line.

For example, given the following invocation of the MaxL Shell,

```
essmsh filename Fiona sunflower
```

and the following subsequent login statement in that session,

```
login $1 identified by $2 on $COMPUTERNAME;
```

- \$COMPUTERNAME is a Windows environment variable.
- \$1 and \$2 refer to the user name and password passed in as arguments at invocation time.

The values of positional parameters can be changed within a session. For example, if the value of \$1 was originally Fiona (because `essmsh` was invoked with `Fiona` as the first argument), you can change it using the following syntax: `set 1 = arg_new;`

Note: If you nest MaxL Shell scripts or interactive sessions, the nested shell does not recognize positional parameters of the parent shell. The nested shell should be passed separate arguments, if positional parameters are to be used.

The file or process that the MaxL Shell reads from can be referred to with the positional parameter \$0. Examples:

- 1) Invocation: `essmsh filename`
\$0 = filename
- 2) Invocation: `program.sh | essmsh -i`
\$0 = stdin
- 3) Invocation: `essmsh`
\$0 = null

Locally Defined Shell Variables

You can create variables of any name in the MaxL Shell without the use of arguments or positional parameters. These variables persist for the duration of the shell session, including in any nested shell sessions.

Example:

```
MaxL>login user1 identified by password1;
MaxL>set var1 = sample;
MaxL>echo $var1; /* see what the value of $var1 is */
sample
MaxL>display application $var1; /* MaxL displays application "sample" */
```

Note: Locally defined variables can be named using alphabetic characters, numbers, and the underscore (`_`). Variable *values* can be any characters, but take note of the usual quoting and syntax rules that apply for the MaxL Shell (see [“MaxL Shell Syntax Rules and Variables” on page 828](#)).

Note: Variables defined or changed in a nested script persist into the parent script after the nested script executes.

Quotation Marks and Variable Expansion

In the following examples, assume you logged in to the MaxL Shell interactively with arguments, as follows. In addition to these examples, see [“Quoting and Special Characters Rules for MaxL Shell” on page 831](#).

```
essmsh -a Fiona sunflower sample basic login $1 $2;
```

Example	Return Value	Explanation
echo \$1;	Fiona	\$1 is expanded as the first invocation argument.
echo "\$1's hat";	Fiona's hat	\$1 is expanded as the first invocation argument, and the special character ' is allowed because double quotation marks are used.
echo \$3;	sample	\$3 is expanded as the third invocation argument.
echo '\$3';	\$3	\$3 is taken literally and not expanded, because it is protected by single quotation marks.
display database \$3.\$4;	Database sample.basic is displayed.	\$3 and \$4 are expanded as the third and fourth invocation arguments. \$3.\$4 is interpreted as two tokens, which makes it suitable for "DBS-NAME" on page 780.
echo "\$3.\$4";	sample.basic, but interpreted as one token (NOT suitable for "DBS-NAME" on page 780, which requires two tokens).	\$3 and \$4 are expanded as the third and fourth invocation arguments, but the entire string is interpreted as a single token, because of the double quotation marks.

Exit Status Variable

A successful MaxL Shell operation should have an exit status of zero. Most unsuccessful MaxL Shell operations have an exit status number, usually 1. Exit status can be referred to from within the shell, using \$? . For example,

```
MAXL> create application test1;
OK/INFO - 1051061 - Application test1 loaded - connection established.
OK/INFO - 1054027 - Application [test1] started with process id [234].
OK/INFO - 1056010 - Application test1 created.
MAXL> echo $?;
0

MAXL> drop application no_such;
ERROR - 1051030 - Application no_such does not exist.
MAXL> echo $?;
2
```

Quoting and Special Characters Rules for MaxL Shell

These rules are for MaxL Shell commands. Applicable MaxL Shell commands include [Spool on/off](#), [Echo](#), [Shell Escape](#), and [Nesting](#).

See Also

[“Quoting and Special Characters Rules for MaxL Language” on page 816](#)

[“Tokens enclosed in Single Quotation Marks” on page 817](#)

[“Tokens Enclosed in Double Quotation Marks” on page 817](#)

[“Use of Backslashes in MaxL” on page 817](#)

[“Use of Apostrophes \(Single Quotation Marks\)” on page 818](#)

Tokens enclosed in single quotation marks

Contents within single quotation marks are preserved as literal, without variable expansion.

Example: `echo '$3'`;

Result: `$3`

Tokens enclosed in double quotation marks

Contents of double quotation marks are treated as a single token, and the contents are perceived as literal except that variables are expanded.

Example: `spool on to "$ESSBASEPATH\out.txt"`;

Result: MaxL Shell session is recorded to `C:\Hyperion\products\Essbase\EssbaseServer\out.txt`.

Example: `spool on to "Ten o'clock.txt"`

Result: MaxL Shell session is recorded to a file named `Ten o'clock.txt`

Use of apostrophes (single quotation marks)

Preserved if enclosed in double quotation marks. Otherwise, causes a syntax error.

Example: `spool on to "Ten o'clock.txt"`

Result: MaxL Shell session is recorded to a file named `Ten o'clock.txt`

Use of Backslashes

Backslashes must be enclosed in single or double quotation marks because they are special characters.

One backslash is treated as one backslash by the shell, but is ignored or treated as an escape character by MaxL. Two backslashes are treated as one backslash by the shell and MaxL.

`'\ ' = \` (MaxL Shell)

`'\ ' = (nothing)` (MaxL)

`'\\ ' = \\` (MaxL Shell)

`'\\ ' = \` (MaxL)

Example: `spool on to 'D:\output.txt'`

Result: MaxL Shell records output to `D:\output.txt`.

Example: `spool on to 'D:\\output.txt'`

Result: MaxL Shell records output to `D:\output.txt`.

Example: `import database sample.basic lro from directory "$ARBORPATH\app\nsample-basic-lros";`

Result: Error. Import is a MaxL statement, and for MaxL, '\ ' is ignored.

Example: `import database sample.basic lro from directory "$ARBORPATH\app\nsample-basic-lros";`

Result: MaxL imports LRO information to Sample Basic from `$ARBORPATH\app\nsample-basic-lros`.

MaxL Shell and Unicode

MaxL Shell is in native mode when started in interactive mode.

MaxL Shell is in native mode when processing a script without a UTF8 byte header.

MaxL Shell is in UTF8 mode when processing a script with the UTF8 byte header.

For more information, see the *Oracle Essbase Database Administrator's Guide* section titled "Compatibility Between Different Versions of Client and Server Software."

MaxL Shell Command Reference

The following topics describe the MaxL Shell commands.

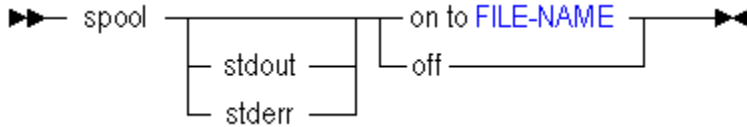
- [Spool on/off](#)
- [Set display column width](#)
- [Set message level](#)
- [Set Timestamp](#)
- [Echo](#)
- [Shell Escape](#)
- [Nesting](#)
- [Error Checking and Branching](#)
- [Cube Deployment from Essbase Studio](#)
- [Version](#)
- [Logout](#)
- [Exit](#)

Spool on/off

Log the output of a MaxL Shell session to a file. Send standard output, informational messages, error messages, and/or warning messages generated by the execution of MaxL statements to a file.

If FILE-NAME does not exist, it is created. If FILE-NAME already exists, it is overwritten. If a directory path is not specified for FILE-NAME, FILE-NAME is created in the current directory of the MaxL Shell. Directories cannot be created using the spool command.

Message logging begins with **spool on** and ends with **spool off**.



Example

```
spool on to 'output.txt';
```

{MaxL statements}

```
spool off;
```

Sends output of MaxL statements to a file called output.txt, located in the current directory where the MaxL Shell was invoked, or in eas\console\bin if the MaxL Script Editor is being used.

```
spool on to 'c:\hyperion\output.txt';
```

Sends output of MaxL statements to a file called output.txt, located in the pre-existing directory specified by an absolute path.

```
spool on to '../../../../../output.txt';
```

Sends output of MaxL statements to a file called output.txt, located in the pre-existing directory specified by a relative path. The file would be located three directories above the current directory, or three directories above eas\console\bin if the MaxL Script Editor is being used.

Description

Most operating systems support three channels for input/output:

- STDIN (standard input channel)
- STDOUT (standard output channel)
- STDERR (standard error channel)

Most operating systems also provide command-line options for re-directing data generated by applications, depending on which of the above channels the data is piped through.

Errors in MaxL are flagged as STDERR, allowing command-line redirection of errors using operating-system redirection handles. Non errors are flagged as STDOUT; thus normal output may be logged separately from error output. Here is an example of redirecting error-output at invocation time:

```
essmsh script.mxl 2>errorfile.err
```

Note: Operating-system redirection handles vary; check the platform documentation.

You can also redirect STDERR and STDOUT independently to different MaxL output logs, using the corresponding options in the `spool` command. For example, you can direct errors to one file and output to another by placing the following lines in your script:

```
spool stdout on to 'output.txt';
spool stderr on to 'errors.txt';
```

or you can direct errors only:

```
spool stderr on to 'errors.txt';
```

or you can direct output only:

```
spool stdout on to 'output.txt';
```

Note: You cannot use the generic `spool` and the special output-channel spools in the same script. For example, the following is not valid:

```
spool on to 'session.txt';
spool stderr on to 'errors.txt';
```

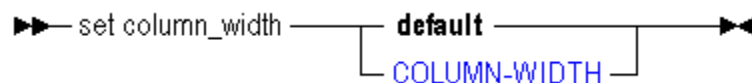
Set Display Column Width

Set the width of the columns that appear in MaxL display output tables, for the current MaxL Shell session.

Default: 20 characters

Minimum: 8 characters

Maximum: No maximum.



Example

```
set column_width 10;
```

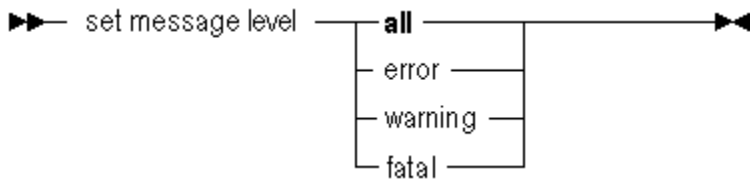
Sets the column width to 10 characters.

```
set column_width default;
```

Sets the column width back to 20 characters.

Set Message Level

Set the level of messaging you want returned from MaxL Shell sessions. By default, all messages are returned.



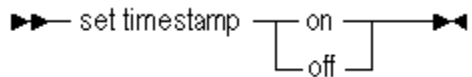
Message level	Description
all	Errors, warnings, status reporting, and informational messages. This is the default message level.
error	Essbase and MaxL Shell error messages.
warning	Essbase warning messages.
fatal	Only errors which cause the shell to disconnect from Essbase.

Example

```
set message level all;
```

Set Timestamp

Enable or disable the display of a timestamp after execution of each MaxL statement. By default, no timestamps are returned.



Notes

The timestamp information does not display after the error-control shell statements goto, iferror, and define.

Example

```
set timestamp on;
```

Echo

Display text or expand variables to the screen or to a log file. When used in scripts with spooling (log-file generation) turned on, echo expands variables in the log file. For interactive sessions, variables are not expanded in the log file; instead, the variable name you typed is recorded (for example, \$1).

Syntax

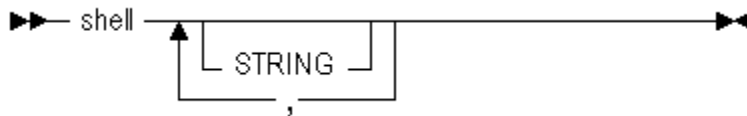
```
echo <text> | <variablename>
```

Example

See examples of echo under the discussion of variables ([“Quotation Marks and Variable Expansion”](#) on page 830).

Shell Escape

Issue operating-system commands directly from a MaxL Shell session. The operating-system output becomes part of the shell session's output, and may be logged to a file. When the operating system finishes executing whatever commands are issued (as STRING), it returns control to the shell session.



Nesting

Reference (include) a MaxL script from within another MaxL script. You might use this if variables are defined in the referenced MaxL script which are useful to the current MaxL script.

Syntax

```
msh <scriptfile>;
```

Example

```
login fiona sunflower;  
alter database sample.basic end archive;  
msh calculate.msh;  
alter database sample.basic  
  begin archive to file bak;  
  logout;
```

Note: Variables defined or changed in a nested script persist into the parent script after the nested script executes.

Note: Because msh is a shell command, it is limited to the originating session. Therefore, you should not reference MaxL scripts that contain new login statements.

Error Checking and Branching

The MaxL Perl Module is the most powerful way to integrate error handling into MaxL. However, the following method is for users who do not implement the MaxL Perl Module.

IfError instructs the MaxL Shell to respond to an error in the previous statement by skipping subsequent statements, up to a certain location in the script that is defined by a label name.

IfError checks the presence of errors only in the precedent statement. IfError checks for:

- Errors in MaxL statement execution
- Errors in MaxL Shell command execution, including:
 - Errors in `spool on/off`, such as permission errors
 - Errors in `set column_width`, such as invalid widths
 - Errors in script nesting, such as permission errors or nonexistent include files

Goto forces the MaxL Shell to branch to a certain location in the script defined by a label name; goto is not dependent on the occurrence of an error.

Syntax

```
iferror LABELNAME
goto LABELNAME
define label LABELNAME
```

Example: Iferror (MaxL)

The following example script contains a dimension build statement and a data load statement. If the dimension build fails, the data load is skipped.

```
login $1 $2;

import database sample.basic dimensions
  from data_file 'C:\\data\\dimensions.txt'
  using rules_file 'C:\\\\data\\rulesfile.rul'
  on error append to 'C:\\\\logs\\dimbuild.log';

iferror 'dimbuildFailed';

import database sample.basic data from data_file
"$ARBORPATH\\app\\sample\\basic\\calcdat.txt"
  on error abort;

define label 'dimbuildFailed';
exit;
```

Example: Iferror (MaxL Shell)

The following example script tests various errors including MaxL Shell errors, and demonstrates how you can set the exit status variable to a nonzero argument to return an exit status to the MaxL Shell.

```
### Begin Script ###

login $1 $2;
echo "Testing syntactic errors...";

spool on to spool.out;

set timestampTypo on;
iferror 'End';
```

```

echo "Testing shell escape...";
shell "cat doesnotexist.txt";
iferror 'ShellError';

msh "doesnotexistlerr.mxl";
iferror 'FileDoesNotExistError';

echo "Script completed successfully...";
spool off;
logout;
exit 0;

define label 'FileDoesNotExistError';
echo "Error detected: Script file does not exist";
spool off;
logout;
exit 1;

define label 'ShellError';
echo ' Shell error detected...';
spool off;
logout;
exit 2;

define label 'End';
echo ' Syntax error detected...';
spool off;
logout;
exit 3;

### End Script ###

```

Example: Goto

The following example script contains a dimension build statement and a data load statement. Goto is used to skip the data load.

```

login $1 $2;

import database sample.basic dimensions
  from data_file 'C:\\data\\dimensions.txt'
  using rules_file 'C:\\\\data\\rulesfile.rul'
  on error append to 'C:\\\\logs\\dimbuild.log';

goto 'Finished';

import database sample.basic data from data_file
"$ARBORPATH\\app\\sample\\basic\\calcdat.txt"
  on error abort;

define label 'Finished';
exit;

```

Notes

The MaxL Shell will skip forward in the script to LABELNAME but not backwards.

Cube Deployment from Essbase Studio

The MaxL Shell Deploy statement replicates the behavior of the Oracle Essbase Studio Cube Deployment Wizard.

For more information, please see [Deploy](#), listed in the MaxL Statements section.

Version

To see which version of MaxL you are using, type `version`.

Example

```
version;
```

Returns

```
Essbase MaxL Shell - Release 11.1.2  
Copyright (c) 2000, 2010, Oracle and/or its affiliates.  
All rights reserved.  
MAXL>
```

Logout

Log out from Essbase without exiting the interactive MaxL Shell.

Syntax

```
logout;
```

Example

```
logout;
```

Exit

Exit from the `MAXL>` prompt after using interactive mode. You can optionally set the exit status variable to a non zero argument to return an exit status to the parent shell.

Note: It is not necessary to exit at the end of MaxL script files or stream-oriented input (using the `-i` switch).

Syntax

```
exit;
```

Example

```
exit;
```

Closes the MaxL Shell window or terminal.

```
exit 10;
```


Closes the MaxL Shell window or terminal with a return status of 10. You can use this in combination with `IfError` to return a non zero error status to the parent shell.

MaxL Perl Module

The MaxL Perl Module, `Essbase.pm`, provides access to Essbase multi-dimensional databases from Perl programs through MaxL, the multi-dimensional access language for Essbase. Communication from Perl to MaxL to Essbase provides the system-administrative functionality of MaxL with the rich programmatic control of Perl.

This section contains the following topics:

- “Installation Help” on page 841
- “Functions” on page 843
- “Perl Scripting Examples” on page 845

To get Perl and learn about it, go to the [Comprehensive Perl Archive Network](#).

Installation Help

The MaxL Perl Module is available for all supported Essbase platforms.

Windows Prerequisites

We recommend that you download the Perl source from www.cpan.org and build it yourself. You may also use a binary distribution; many of these are listed on www.cpan.org.

Before you install the `Essbase.pm` extension to Perl, ensure that:

1. You have Perl 5.6 (or higher) installed on your system.
2. You have Microsoft Visual C++ version 6 or higher installed on your system.
3. The Essbase Server is either installed locally, or you have at least the Runtime Client installed and your system's environment is set up to access a remote Essbase Server. Your system should have an environment variable `$ESSBASEPATH` pointing to the root directory of the Essbase Server installation. In addition, `%ESSBASEPATH%\Bin` should be included in your path variable.

Note: MaxL Perl Module can only be used with the same version Essbase Server.

Windows Instructions

1. Install Essbase Server. The MaxL Perl Module files are included as part of the installation, and a `Perlmod` directory will be created under `%ESSBASEPATH%`.
2. Follow the instructions in `README`, included in the `Perlmod` directory.

UNIX Prerequisites

Before you install the `Essbase.pm` extension to Perl, ensure that:

1. You have Perl 5.6 (or higher) installed on your system.
2. You have a C compiler installed on your system.
3. The Essbase Server is installed. Your system should have an environment variable `$ESSBASEPATH` pointing to the root directory the Essbase installation. In addition, `$ESSBASEPATH/bin` should be included in your path variable.
4. The following MaxL and Essbase files exist in the appropriate directories. If Essbase is installed correctly, this is already the case.

File Name	Directory
<code>essmsh</code>	<code>\$ESSBASEPATH/bin</code>
<code>essmaxl.h</code>	<code>\$ESSBASEPATH/api/include</code>
<code>maxldefs.h</code>	<code>\$ESSBASEPATH/api/include</code>
<code>essapi.h</code>	<code>\$ESSBASEPATH/api/include</code>
<code>essxlat.h</code>	<code>\$ESSBASEPATH/api/include</code>
<code>esstypes.h</code>	<code>\$ESSBASEPATH/api/include</code>
<code>esstsa.h</code>	<code>\$ESSBASEPATH/api/include</code>
<code>essauth.h</code>	<code>\$ESSBASEPATH/api/include</code>
<code>libessutlu.so</code>	<code>\$ESSBASEPATH/bin</code>
<code>libessshru.so</code>	<code>\$ESSBASEPATH/bin</code>
<code>libessotlnu.so/libessotlsu.so</code>	<code>\$ESSBASEPATH/bin</code>
<code>libesssdapiu.so/libessdvrq.so</code>	<code>\$ESSBASEPATH/bin</code>
<code>libglobalc.so</code>	<code>\$ESSBASEPATH/bin</code>

Note: You do not have to install the API to use MaxL. The necessary `api/include` and `api/lib` directories are created to contain the MaxL; libraries and header files.

UNIX Instructions

1. If you have met the above prerequisites, change to the MaxL Perl Module; directory, which is `perlmod` in the Essbase directory.
2. Follow the instructions in README, included in the `perlmod` directory.

Functions

- “connect (*user*, *password*, *host*);” on page 843
- “do (*statement*);” on page 843
- “pop_msg();” on page 843
- “fetch_desc();” on page 844
- “fetch_row();” on page 845
- “disconnect();” on page 845

connect (*user*, *password*, *host*);

- *user*- Required. The Essbase user name.
- *password*- Required. A valid password for *user*.
- *host*- Optional. The computer name hosting the Essbase instance.

Usage

```
my $dbh = Essbase->connect ("user", "password", "host");
```

Establishes a connection to Essbase using \$dbh, the database handle in "my" namespace. Returns: A session object (for example, \$dbh).

do (*statement*);

- *statement*- Required. A MaxL statement to be passed to the Essbase Server.

Usage

```
$dbh->do ("display user");
```

Where **display user** is a valid MaxL statement.

Returns (and sets Essbase{STATUS} to):

\$MAXL_STATUS {NOERR} if execution was successful. There are likely to be informational and feedback messages on the message stack, which may be obtained with pop_msg().

\$MAXL_STATUS {ERROR} if there was a user error. Error numbers, levels, and texts may be obtained with the pop_msg method.

Note: There are likely to be informational messages on the message stack even if execution was successful. These also may be obtained using pop_msg.

pop_msg();

Navigates through MaxL status messages one at a time.

Arguments: none.

Returns: a list of the form (*<message_number>*, *<message_level>*, *<message_text>*)

Each invocation of the "do" method results in a stack of status messages. This stack is unwound by repeatedly calling `pop_msg` until it returns nothing. It is acceptable for a Perl program to ignore the message stack or to unwind it only partially. The next call to "do" will clear left-over messages.

There will probably be a number of messages on the stack even after a successful execution. In most cases, a Perl program will only need to know if the execution of the last "do" was successful, which is indicated by the return value from "do".

When the message stack is empty, the return list elements are undefined and `Essbase{STATUS}` is set to `$MAXL_STATUS{END_OF_DATA}`.

fetch_desc();

Returns a reference to a row of query results and a reference to a corresponding row of datatypes for the query results.

The function should be called as follows:

To return column names and datatypes:

```
($column_name, $datatypes) = $dbh->fetch_desc();
```

To return only column names:

```
($column_name) = $dbh->fetch_desc();
```

A datatype is information about what kind of data a particular value is. For example, `Hello` is a string, and is represented by a `Char` datatype. `0` could be a `Number`, but it could also be a `False` value for a `Boolean` datatype.

If you fetch only column-description records and ignore the datatypes, the array of values might look like the following:

```
application  comment  startup  max_file_size
```

By fetching the datatype information in addition to the column values, the array of values might look like the following:

```
application  comment  startup  max_file_size
           3           3           1           2
```

A row of datatype is defined the same way as a row of column descriptions: { `val[0]`, `val[1]`, ..., `val[NUM_OF_FIELDS-1]` }

Row numbers are counted cardinally from 0: [0, 1, 2, ..., NUM_OF_ROWS - 1]

The values placed into the row of datatypes are 0, 1, 2, or 3 corresponding to the values of `MAXL_DTINT_T` inside `maxldefs.h`.

```
None = 0
Bool = 1
Number = 2
Char = 3
```

fetch_row();

Returns a reference to a row of query results in a MaxL output table, as a list.

Essbase->{STATUS} is set to one of the following:

- \$MAXL_STATUS{NOERR} on success.
- \$MAXL_STATUS{END_OF_DATA} if there were no rows to fetch.
- \$MAXL_STATUS{ERROR} if a user error has occurred.

A row of record is defined as { val[0], val[1], ... , val[NUM_OF_FIELDS-1] } }

Row numbers are counted cardinally from 0:[0, 1, 2, ... , NUM_OF_ROWS - 1]

disconnect();

Terminates an Essbase session and destroys the session object.

Returns: Completion status.

Perl Scripting Examples

Createuser.pl

The following is the simplest example of a Perl script using `Essbase.pm`. The script establishes a connection to the Essbase Server, creates a user, and disconnects.

```
# Use the Essbase.pm module. This statement is required to use Essbase within
a Perl script.
use Essbase;

# Create a handle to the Essbase Server by connecting as admin, mypassword to
the local machine.
my $dbh = Essbase->connect("admin", "mypassword", "localhost");

# Use the do Perl function to pass the MaxL create user statement (enclosed
in quotation marks) to the Essbase Server.
$dbh->do("create user Essbase identified by mypassword");

# Disconnect from the Essbase Server.
$dbh->disconnect();
```

Createusers.pl

The following Perl script tests whether Perl is able to use the MaxL Perl Module. If `Essbase.pm` is loaded, the program establishes a connection to Essbase, creates three users with different passwords, and disconnects.

```
##### print on failure.

BEGIN { $| = 1; }
END {print "ERROR: System NOT Loaded\n" unless $loaded;}
use Essbase;
```

```

$loaded = 1;

#####

sub create_user
{
# In connect statements, replace the sample login details.
  my $dbh = Essbase->connect("admin", "pass1", "localhost");

# Create array of users.
  @user = (
    "Fred",
    "George",
    "Mary",
  );

# Create array of passwords.
  @password = (
    "password1",
    "password2",
    "password3",
  );

  $i = 0;

  while ($i le 2) {

    $username = $user[$i];
    $newpassword = $password[$i];
    $j = $i + 1;

    print $dbh->do("create user $username identified by $newpassword") == 0 ? "user$j
created\n" : "ERROR: user user$j NOT created\n";

    $i = $i + 1;

  }

  print $dbh->disconnect() == 0 ? "Essbase database handle released\n" : "ERROR:
Essbase database handle NOT released\n";
}

#
# Create user test.
#
&create_user;

```

Maketable.pl

The following subroutines from a Perl script return a message list that resulted from executing a MaxL statement, and build a table from a result set.

```

use Essbase;

#
# Returns a message list that resulted from executing

```

```

# a MaxL statement.
#
sub msgs
{
    my $dbh = shift(@_);
    my $msglist;

    # dump all messages one thread at a time
    while (1)
    {
        my ($msgno, $level, $msg);

        ($msgno, $level, $msg) = $dbh->pop_msg();
        # gets us out of the loop if a $msg comes back as undef
        last if ! $msg;
        $msgstr = sprintf " %-8d", $msgno;
        $msglist .= "$msgstr - $msg\n";
    }

    return $msglist;
}

#
# Returns a result set in the form of a table.
#
sub tab
{
    my $dbh = shift;
    my ($colnum, $rec, $dt, $name, $tab, $line);

    # build an output table

    # setup the header
    ($name, $dt) = $dbh->fetch_desc();
    for ($col = 0; $col < $dbh->{NUM_OF_FIELDS}; $col++)
    {
        $str = sprintf " %-19.19s", $name->[$col];
        $tab .= $str;
        $line .= "+-----";
    }

    $tab .= "\n$line\n";

    # now populate the table with data
    $rec = $dbh->fetch_row();
    while(defined($rec))
    {
        for ($col = 0; $col < $dbh->{NUM_OF_FIELDS}; $col++)
        {
            if ($dt->[$col] == 3) {
                #format for characters
                $str = sprintf " %-19.19s", $rec->[$col];
            } elsif ($dt->[$col] == 2) {
                #format for numbers
                $str = sprintf " %19.19s", $rec->[$col];
            } elsif ($dt->[$col] == 1) {
                #format for bools
            }
        }
    }
}

```

```

        if ($rec->[$col] == 0) {
            $str = sprintf " %19.19s", "FALSE";
        } else {
            $str = sprintf " %19.19s", "TRUE";
        }
    }
    $stab .= $str;
}
$stab .= "\n";
$rec = $dbh->fetch_row();
}
$stab .= "\n";

if ($stab =~ s/^\n//)
{
    $stab="";
}

return $stab;
}

```

ESSCMD Script Conversion

`cmd2mxl` is a fully supported utility for converting existing ESSCMD scripts to their corresponding MaxL scripts. To convert an ESSCMD script to a MaxL script, go to the operating-system command prompt and enter the executable name, the ESSCMD script name, the desired MaxL script name, and the name of a logfile to write to in case of errors.

- [“ESSCMD Script Utility Usage” on page 848](#)
- [“Things to Note About the ESSCMD Script Utility” on page 848](#)
- [“ESSCMD to MaxL Mapping” on page 849](#)

ESSCMD Script Utility Usage

```
cmd2mxl esscmd_script maxl_output logfile
```

For example, if the ESSCMD script name is `%ARBORPATH%\dailyupd.scr`, the command issued on the operating-system command line would be:

```
cmd2mxl %ARBORPATH%\dailyupd.scr %ARBORPATH%\dailyupd.mxl %ARBORPATH%
\log\dailyupd.log
```

Subsequently, the MaxL script can be executed using the MaxL Shell by the following command:

```
essmsh %ARBORPATH%\dailyupd.mxl
```

Things to Note About the ESSCMD Script Utility

1. The utility will only translate syntactically and semantically valid ESSCMD scripts.
2. For invalid ESSCMD scripts, the resulting MaxL script is undefined.

3. All ESSCMD statements in the scripts should end with a semicolon (;) statement terminator.
4. This utility will only work on Windows platforms.
5. Although most ESSCMD commands have corresponding MaxL statements, there are exceptions. For such exceptions, a comment will be generated in the logfile, and the resulting MaxL script will have to be modified to work correctly. Note that if an ESSCMD command is still needed, it can be invoked from a MaxL script using `shell esscmd <scriptname>`.
6. All strings in the ESSCMD scripts should be surrounded by double quotation marks (").

ESSCMD to MaxL Mapping

The following table compares ESSCMD usage to MaxL usage, and the following conversions are supported by `cmd2mxl`.

ESSCMD Command	ESSCMD Usage Example	MaxL Equivalent Example
ADDUSER	ADDUSER finance essexer1;	alter user essexer1 add to group finance;
BEGINARCHIVE	beginarchive sample basic "test.txt";	alter database Sample.Basic begin archive to file 'test.txt';
BEGININCBUILDDIM	beginincbuilddim;	import database Sample.Basic dimensions from local text data_file 'c:\\data.txt' using local rules_file 'c:\\data_rule.ru' on error write to 'c:\\error.log';
BUILDDIM	builddim 1 "c:\\data_ru.ru" 3 "c:\\data.txt" 4 "c:\\error.log";	Same as BEGININCDIMBUILD
CALC	calc "CALC ALL;";	execute calculation 'CALC ALL' on sample.basic;
CALCDEFAULT	calcdefault;	execute calculation default on Sample.Basic;
CALCLINE	calcline "CALC ALL;";	execute calculation 'CALC ALL;' on sample.basic;
COPYAPP	copyapp sample sampnew;	create application sampnew as sample;
COPYDB	copydb sample basic sample basic2;	create or replace database sample.basic2 as sample.basic;
COPYFILTER	copyfilter sample basic westwrite sample basic westmgr;	create filter sample.basic.westmgr as sample.basic.westwrite;
COPYOBJECT	copyobject "9" "sample" "basic" "calcdat" "sample" "basic" "calcdat2";	alter object sample.basic.calcdat of type text copy to 'sample.basic.calcdat2';
CREATEAPP	createapp finance;	create or replace application finance;
CREATEDB	createdb finance investor;	create or replace database finance.investor;
CREATEGROUP	creategroup managers;	create group managers;

ESSCMD Command	ESSCMD Usage Example	MaxL Equivalent Example
CREATELOCATION	select sample basic; createlocation hq hqserver finance investor admin password;	alter system load application sample; alter application sample load database basic; create location alias hq from sample.basic to finance.investor at hqserver as admin identified by 'password';
CREATEUSER	createuser karen password;	create user karen identified by 'password';
CREATEVARIABLE	createvariable CurMnth localhost sample basic Jan;	alter database sample.basic add variable CurMnth 'Jan'; alter application sample add variable CurMnth 'Jan'; alter system add variable CurMnth 'Jan';
DELETEAPP	deleteapp sampnew;	drop application sampnew cascade;
DELETEDB	deletedb demo basic;	drop database demo.basic;
DELETEGROUP	deletegroup engg;	drop group engg;
DELETELOCATION	select finance investor; deletelocation hq1;	alter system load application finance; alter application finance load database investor; drop location alias finance.investor.hq1;
DELETELOG	deletelog sample;	alter application sample clear logfile;
DELETEUSER	deleteuser rob;	drop user rob;
DELETEVARIABLE	select sample basic; deletevariable CurMnth "localhost";	alter system load application sample; alter application sample load database basic; alter database sample.basic drop variable CurMnth; alter application sample drop variable CurMnth; alter system drop variable CurMnth;
DISABLELOGIN	disablelogin demo;	alter application demo disable connects;
DISPLAYALIAS	select sample basic; displayalias "default";	query database sample.basic list alias_names in alias_table 'Default';
ENABLELOGIN	enablelogin demo;	alter application demo enable connects;
ENDARCHIVE	endarchive sample basic;	alter database sample.basic end archive;
ENDINCBUILDDIM	ENDINCBUILDDIM;	See BEGININCBUILDDIM
ESTIMATEFULLDBSIZE	select sample basic; estimatefulldbsize;	query database sample.basic get estimated size;
EXIT	exit;	exit;

ESSCMD Command	ESSCMD Usage Example	MaxL Equivalent Example
EXPORT	select sample basic; export "c:\data.txt" 1;	alter system load application sample; alter application sample load database basic; export database Sample.Basic all data to data_file 'c:\\data.txt';
GETALLREPLCELLS	select samppart company; getallreplcells "svr2" "sampeast" "east";	alter system load application samppart; alter application samppart load database company; refresh replicated partition samppart.company from sampeast. east at svr2;
GETAPPINFO	getappinfo "demo";	display application demo;
GETAPPSTATE	getappstate demo;	display application demo;
GETATTRIBUTESPECS	select sample basic; getattributespecs;	query database sample.basic get attribute_spec;
GETATTRINFO	select sample basic; getattrinfo "Caffeinated_True";	query database sample.basic get attribute_info 'Caffeinated_ True';
GETDBINFO	select sample basic; getdbinfo;	display database sample.basic request_history;
GETDBSTATE	getdbstate sample basic;	display database sample.basic;
GETDBSTATS	select sample basic; getdbstats;	query database sample.basic get dbstats data_block;
GETCRRATE	getcrrate;	query database sample.basic get currency_rate;
GETDEFAULTCALC	select sample basic; getdefaultcalc;	query database sample.basic get default calculation;
GETMBRCALC	select sample basic; getmbrcalc "Profit %";	query database sample.basic get member_calculation 'Profit %';
GETMBRINFO	select sample basic; getmbrinfo "Ounces_20";	query database sample.basic get member_info 'Ounces_20';
GETPERFSTATS	select sample basic; getperfstats;	query database sample.basic get performance statistics kernel_ cache table;
GETUPDATEDREPLCELLS	See GETALLREPLCELLS	See GETALLREPLCELLS
GETUSERINFO	getuserinfo admin;	display user admin;
GETVERSION	getversion;	version;

ESSCMD Command	ESSCMD Usage Example	MaxL Equivalent Example
IMPORT	select sample basic; import 1 "c:\data.txt" 4 y 3 "c:\import.rul" n "c:\data_load.err";	alter system load application sample; alter application sample load database basic; import database sample.basic data from local text data_file 'c:\data.txt' using local rules_file 'c:\\data_rule.rul' on error write to 'c:\\data_load.err';
INCBUILDDIM	See BEGININCBUILDDIM	See BEGININCBUILDDIM
LISTALIASES	select sample basic; listaliases;	query database sample.basic list alias_table;
LISTAPP	listapp;	display application all;
LISTDB	listdb;	display database all;
LISTFILES	listfiles "" "sample" "basic";	query database sample.basic list all file information;
LISTFILTERS	listfilters sample basic;	display filter on database Sample.Basic;
LISTGROUPS	listgroups;	display group all;
LISTGROUPUSERS	listgroupusers finance;	display user in group finance;
LISTLINKEDOBJECTS	select sample basic; listlinkedobjects "Fiona" "07/07/2003";	query database sample.basic list Iro by Fiona before '07/07/2003';
LISTLOCATIONS	select sample basic; listlocations;	alter system load application sample; alter application sample load database basic; display location alias on database sample.basic;
LISTLOCKS	listlocks;	display lock;
LISTLOGINS	listlogins;	display session all;
LISTOBJECTS	listobjects "2" "Sample" "Basic";	display object of type calc_script on database sample.basic;
LISTUSERS	listusers;	display user all;
LISTVARIABLES	listvariables localhost sample basic;	display variable on database sample.basic;
LOADALIAS	select sample basic; loadalias "special_flavors" "C:\Hyperion\products\Essbase\EssbaseServer\app\sample\basic\seasonal.txt";	alter database sample.basic load alias_table 'special_flavors' from data_file "\$ARBORPATH\app\sample\basic\seasonal.txt";
LOADAPP	loadapp sample;	alter system load application sample;
LOADDB	loaddb sample basic;	alter application sample load database basic;

ESSCMD Command	ESSCMD Usage Example	MaxL Equivalent Example
LOADDATA	select sample basic; loaddata 3 "c:\data.txt";	alter system load application sample; alter application sample load database basic; import database sample.basic data from local text data_file 'c:\data.txt' on error abort;
LOGIN	login local admin password;	login admin 'password' on local;
LOGOUT	logout;	logout;
LOGOUTALLUSERS	logoutallusers y;	alter system logout session all;
LOGOUTUSER	Available only in interactive ESSCMD sessions.	alter system logout session 4294967295;
OUTPUT	output 1 c:\test.log; output 4;	spool on to 'c:\test.log'; spool off;
PURGELINKEDOBJECTS	purgelinkedobjects "Fiona" "07/07/2002";	alter database sample.basic delete lro by 'fiona' before '07/07/2002';
PUTALLREPLCELLS	select sampeast east; putallreplcells svr1 samppart company;	alter system load application sampeast; alter application sampeast load database east; refresh replicated partition sampeast.east from samppart. company at svr1 updated data;
PUTUPDATEDREPLCELLS	See PUTALLREPLCELLS	See PUTALLREPLCELLS
REMOVELOCKS	removelocks "2";	drop lock held by Fiona;
REMOVEUSER	removeuser finance steve;	alter user steve remove from group finance;
RENAMEAPP	renameapp sample newsamp1;	alter application sample rename to newsamp1;
RENAMEDB	renamedb sample basic newbasic;	alter database sample.basic rename to newbasic;
RENAMEFILTER	renamefilter sample basic westmgr allwest;	create or replace filter sample.basic.westmgr as sample.basic. allwest; drop filter sample.basic.westmgr;
RENAMEOBJECT	RENAMEOBJECT "9" "sample" "basic" "calcdat" "calcdat2";	alter object sample.basic.calcdat of type text rename to 'calcdat2';
RENAMEUSER	renameuser steve_m m_steve;	alter user steve_m rename to m_steve;
RESETDB	select sample basic; resetdb;	alter database sample.basic reset;
RESETPERFSTATS	resetperfstats enable;	alter database sample.basic set performance statistics enabled;

ESSCMD Command	ESSCMD Usage Example	MaxL Equivalent Example
RUNCALC	The only command supported is the server based calc script execution. Select Sample Basic; Runcalc 2 one;	execute calculation Sample.Basic.one;
RUNREPT	select sample basic; runrept 2 complex "c:\complex.out";	alter system load application sample; alter application load database basic; export database sample.basic using server report_file 'complex' to data_file 'c:\complex.out';
SELECT	select sample basic;	alter system load application sample; alter application load database basic;
SETALIAS	select sample basic; setalias "long names";	alter database sample.basic set active alias_table 'Long Names';
SETAPPSTATE	setappstate sample "" y y 4 y y y y 1000 1000;	alter application sample enable startup; alter application sample enable autostartup; alter application sample set minimum permission manager; alter application sample enable connects; alter application sample enable commands; alter application sample enable updates; alter application sample enable security; alter application sample set lock_timeout after 1000 seconds; alter application sample set max_lro_file_size 1000 kb;
SETDBSTATE	setdbstate "" "Y" "Y" 4 3145728 "Y" "Y" "Y" "" "" 0 1048576 1025 "Y";	alter database sample.basic enable startup; alter database sample.basic enable autostartup; alter database sample.basic set minimum permission manager; alter database sample.basic set data_cache_size 3145728; alter database sample.basic enable aggregate_missing; alter database sample.basic enable two_pass_calc; alter database sample.basic enable create_blocks; alter database sample.basic set currency_conversion division; alter database sample.basic set index_cache_size 1048576; alter database sample.basic enable compression;
SETDBSTATEITEM	.	See the alter database statement.
SETDEFAULTCALC	select sample basic; setdefaultcalc "CALC ALL";	alter database sample.basic set default calculation as 'CALC ALL';

ESSCMD Command	ESSCMD Usage Example	MaxL Equivalent Example
SETDEFAULTCALCFILE	select sample basic; setdefaultcalcfile defcalc;	Create a calculation file in the server containing the calculation string. Then, alter database sample.sasic set default calculation sample.basic.defcalc; will set the default calculation.
SETMSGLEVEL	setmsglevel 2;	set message level all; Note: This is part of the separate MaxL Shell grammar, not the MaxL language itself. You can use set message level with the MaxL Shell, but it is not embeddable in Perl.
SETPASSWORD	setpassword steve newpass;	alter user steve set password newpass;
SHUTDOWNSERVER	shutdownserver local admin password;	login admin 'password' on local; alter system shutdown;
SLEEP	sleep 10;	shell sleep 10;
UNLOADALIAS	select sample basic; unloadalias "flavors";	alter database sample.basic unload alias_table 'flavors';
UNLOADAPP	unloadapp sample;	alter system unload application sample;
UNLOADDB	unloaddb sample basic;	alter application sample unload database basic;
UNLOCKOBJECT	unlockobject "1" "sample" "basic" "basic";	alter object 'sample.basic.basic' of type outline unlock;
UPDATE	select sample.basic update "Jan Sales '100-10' Florida Actual 220";	import database sample.basic from data_string 'Jan Sales 100-10 Florida Actual 220';
UPDATEFILE	updatefile 3 "c:\data.txt" 1;	same as LOADDATA;
UPDATEVARIABLE	updatevariable hot_product local sample basic "100-10";	alter system set variable 'hot_product' '100-10'; alter application sample set variable 'hot_product' "100-10"; alter database Sample.Basic set variable 'hot_product' '100-10';
VALIDATE	validate;	alter database sample.basic validate data to local logfile 'validation.txt';

Reserved Words List

The following keywords are part of the MaxL DDL grammar, and are reserved. If you intend to use any of these words as names or passwords, you must enclose the word in single quotation marks.

```

abort
absolute_value
account_type
active
add

```

administrator
advanced
after
aggregate
aggregates
aggregate_assume_equal
aggregate_missing
aggregate_storage
aggregate_sum
aggregate_view
aggregate_use_last
algorithm
alias
alias_names
alias_table
all
all_users_groups
allocation
alloc_rule
allow
allow_merge
alter
alternate_rollups
amount
amountcontext
amounttimespan
any
append
application
application_access_type
apply
archive
archive_file
area
as
aso_level_info
at
attribute
attribute_calc
attribute_info
attribute_spec
attribute_to_base_member_association
auto_password
autostartup
b
backup_file
based
basis
basistimespan
basistimespanoptions
before
begin
bitmap
blocks
buffer_id
buffered
build

by
cache_pinning
cache_size
calc_formula
calc_script
calc_string
calculation
cascade
cell_status
change_file
clear
client
cnt_semaphore
column_width
columns
combinebasis
commands
comment
commitblock
committed_mode
compact
compression
compression_info
config_values
connect
connects
consolidation
copy
copy_subvar
copy_useraccess
create
create_application
create_blocks
create_user
creation
creation_user
creditmember
cube_size_info
currency
currency_category
currency_conversion
currency_database
currency_member
currency_rate
custom
data
data_block
data_cache_size
data_file
data_file_cache_size
data_storage
data_string
database
database_synch
database_asynch
days
dbstats

debitmember
debug
default
definition_only
definitions
delete
designer
destroy
dimension
dimensions
direct
direction
directory
disable
disabled
disallow
discard_errors
disk
display
divideamount
division
drillthrough
dml_output
drop
dump
dynamic_calc
eas_loc
enable
enabled
encrypted
end
end_transaction
enforce
eqd
error
error_file
errors_to_highest
errors_to_location
errors_to_lowest
estimated
event
exact
excel
exceeds
excludedrange
execute
existing_views
export
export_directory
external
failed_sss_migration
fragmentation_percent
freespace
from
file
file_location
file_size

file_type
filter
filter_access
fixed_decimal
for
force
force_dump
formatted_value
function
gb
get
get_missing_cells
get_meaningless_cells
global
grant
group
group_id
ha_trace
held
high
hostname
identified
identify
ignore_missing_values
ignore_zero_values
immediate
implicit_commit
import
in
inactive
inactive_user_days
including
incremental
index
index_cache_size
index_data
index_page_size
information
initialize
input
instead
invalid_block_headers
invalid_login_limit
io_access_mode
kb
kernel_io
kernel_cache
kill
level
level0
license_info
linked
list
load
load_buffer
load_buffers
load_buffer_block

local
location
lock
lock_timeout
locked
log_level
logfile
login
logout
long
lotus_2
lotus_3
lotus_4
low
lro
macro
manager
mapped
max_disk_size
max_file_size
max_lro_file_size
mb
medium
member
member_alias_namespace
member_calculation
member_comment
member_data
member_fixed_length_data
member_formula
member_info
member_name_namespace
member_property
member_uda
member_uda_namespace
member_variable_length_data
merge
meta_read
metadata_only
migr_modified_access
miner
minimum
mining
minutes
missing_value
mode
model
move
multiple
multiplication
mutex
name
negativebasisoptions
never
no_access
none
non_unique_members

nonunicode_mode
note
nothing
numerical_display
object
objects
of
off
offset
on
only
opg_cache
opg_state
optional
optional_group
options
or
outline
outline_id
outline_paging_file
output
override
overview
partition
partition_file
partition_size
passive
password
password_reset_days
performance
permission
persistence
perspective
physical
pmml_file
ports
pov
pre_image_access
precision
preserve
preserve_groups
private
privilege
process
project
property
protocol
purge
query
query_data
query_tracking
range
read
recover
reference_cube
reference_cube_reg
refresh

region
registration
reregister
remote
remove
remove_zero_cells
rename
repair
repeatamount
replace
replay
replicated
replication_assume_identical
report_file
request
request_history
request_id
reset
resource_usage
restore
restructure
result
resync
retrieve_buffer_size
retrieve_sort_buffer_size
reverse
revoke
rle
round
row
rows
rules_file
runtime
runtime_info
save
scientific_notation
scope
score
script_file
seconds
security
security_backup
select
selecting
selection
self_session_info
semaphore
sequence_id_range
server
server_port
session
session_idle_limit
session_idle_poll
set
shared_services_native
short
shutdown

single
singlecell
size
size_limit
skip_to_next_amount
skip_missing
skip_negative
skip_zero
slice
sourceregion
spec
spinlock
splitbasis
spread
SSL
sss
sss_mode
sss_name
starting
startup
statistics
status
stop
stopping
storage
storage_info
structure_file
subtract
supervisor
suppress
sync
system
table
tablespace
target
targettimespan
targettimespanoptions
task
tb
template
text
thread
to
total_size
transactions
transformation
transparent
trigger
trigger_func
trigger_spool
two_pass_calc
type
uda
unicode
unicode_mode
unlimited
unload

unlock
update
updated
updates
use
user
username_as_password
using
validate
values
variable
vector
verification
version
view_file
views
volume
wait_for_resources
warn
when
with
wizard
worksheet
write
xml_file
zero_value
zeroamountoptions
zerobasisoptions
zlib

MaxL Statements (Aggregate Storage)

[Click here for non-aggregate storage list](#)

Some MaxL grammar is applicable only to aggregate storage mode, and some standard grammar is not applicable to aggregate storage mode. The following statements support aggregate storage application and database operations.

- [alter application](#)
- [alter database](#)
- [alter filter](#)
- [alter group](#)
- [alter object](#)
- [alter partition](#)
- [alter system](#)
- [alter tablespace](#)
- [alter trigger](#)
- [alter user](#)
- [create application](#)

- create database
- create filter
- create group
- create outline
- create partition
- create after-update trigger
- create user
- display application
- display calculation
- display database
- display disk volume
- display filter
- display filter row
- display group
- display lock
- display object
- display partition
- display privilege
- display session
- display system
- display tablespace
- display trigger
- display user
- display variable
- drop application
- drop calculation
- drop database
- drop filter
- drop group
- drop lock
- drop object
- drop partition
- drop trigger
- drop user
- execute aggregate process

- [execute aggregate build](#)
- [execute aggregate selection](#)
- [export data](#)
- [grant](#)
- [import data](#)
- [import dimensions](#)
- [login](#)
- [query application](#)
- [query database](#)
- [refresh outline](#)
- [refresh replicated partition](#)

The MaxL grammar is case-insensitive. Semicolon statement-terminators are required when using the MaxL Shell. However, do not use semicolons at the end of statements passed using Perl functions. Key words of the MaxL grammar are represented in this document in lower-case. Terminals, represented in upper-case, are to be replaced by the appropriate names, numbers, privileges, or strings. For more information about components of MaxL statements, see “[MaxL Definitions](#)” on page 767.

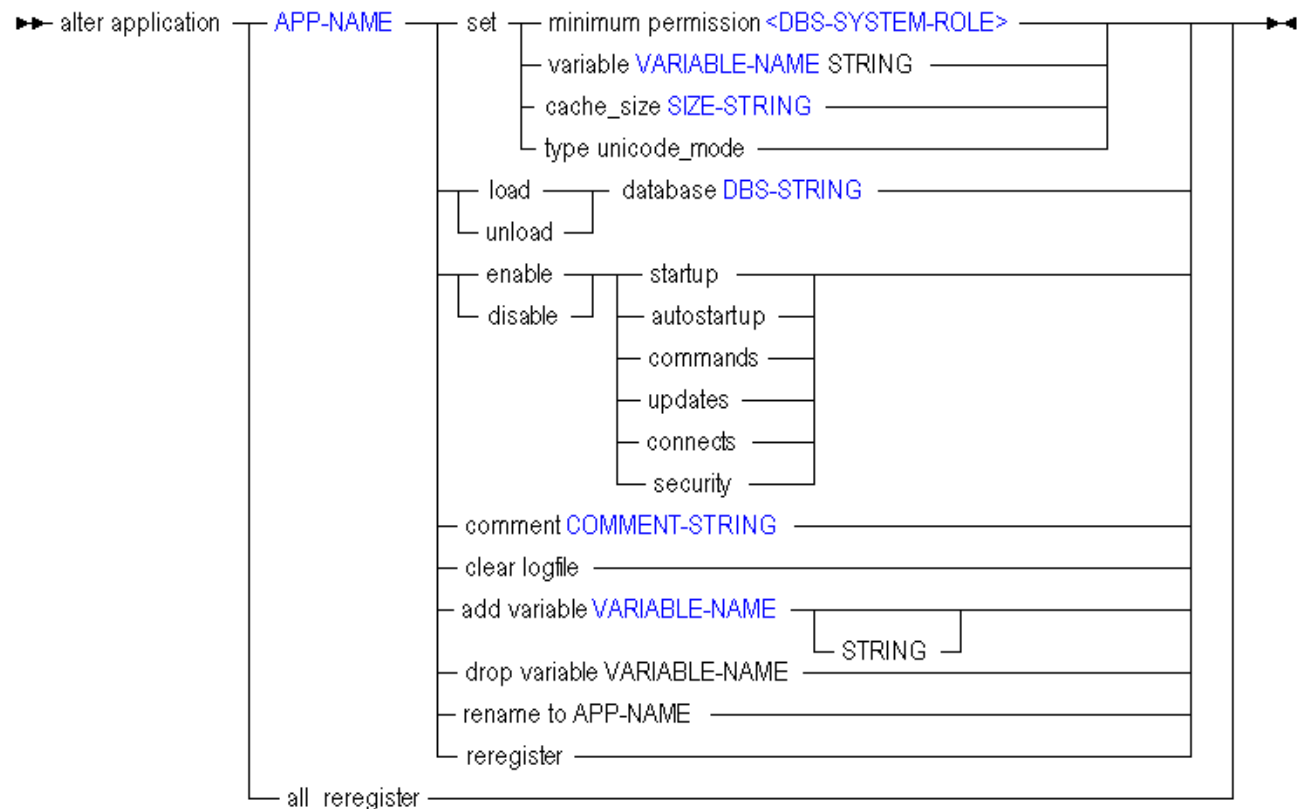
Note: “[Login](#)” on page 826 is part of the separate command shell grammar, not the MaxL language itself. You can use the login statement with the MaxL Shell, but it is not embeddable in Perl. For Perl, use “[connect \(user, password, host\);](#)” on page 843.

Alter Application (Aggregate Storage)

[Click here for non-aggregate storage version](#)

Change application-wide settings. Permission required: Application Manager.

Syntax



You can change the following application-wide settings using **alter application**.

Keyword	Description
set minimum permission	Grant all users a minimum level of permission to all databases in the application. Users with higher permissions than this minimum are not affected.
set variable	Assign a string value to an existing substitution-variable name. If the variable does not exist, first create it using add variable . Substitution variables may be referenced by calculations in the application.
set cache_size	Set the maximum size to which the aggregate storage cache may grow. The aggregate storage cache grows dynamically until it reaches this limit. This setting takes effect after you restart the application. To check the currently set limit, use the following MaxL statement: <pre>query application APP-NAME get cache_size;</pre>
set type unicode_mode	Migrate an application to Unicode mode. Migration to Unicode mode cannot be reversed.
load database	Start (by loading into memory) an idle database. The statement will fail if you do not have at least read privilege for the database.
unload database	Stop (by unloading from memory) an active database. The statement will fail if you do not have at least read privilege for the database.
enable startup	Permit all users to load (start) the application. This only applies to users who have at least read privilege for the application. Startup is enabled by default.

Keyword	Description
disable startup	Prevent all users from loading (starting) the application. Startup is enabled by default.
enable autostartup	Start the application automatically when Essbase Server starts. By default, autostartup is disabled.
disable autostartup	Do not start the application automatically when Essbase Server starts. By default, autostartup is disabled.
enable commands	Allow all users with sufficient permissions to make requests to databases in the application. Use to reverse the effect of disable commands . The disable commands setting remains in effect only for the duration of your session. By default, commands are enabled.
disable commands	Prevent all requests to databases in the application, including non-data-specific requests, such as viewing database information or changing database settings. All users are affected, including other administrators. Administrators are affected by this setting as a safety mechanism to prevent accidental updates to databases during maintenance operations. This setting remains in effect only for the duration of your session. The setting takes effect immediately, and affects users who are currently logged in, as well as users who log in later during your session.
	<hr/> <p>Caution! If performing maintenance operations that require disabling commands, you must make those maintenance operations within the same session and the same script as the one in which commands were disabled.</p> <hr/>
	By default, commands are enabled.
enable updates	Allow all users with sufficient permissions to make requests to databases in the application. Use to reverse the effect of disable updates . Disabling updates remains in effect only for the duration of your session. By default, updates are enabled.
disable updates	Prevent all users from making requests to databases in the application. Use before performing update and maintenance operations. The disable updates setting remains in effect only for the duration of your session.
	<hr/> <p>Caution! If performing maintenance operations that require updates to be disabled, you must make those maintenance operations within the same session and the same script as the one in which updates were disabled. By default, updates are enabled.</p> <hr/>
enable connects	Allow all users with sufficient permissions to make connections to databases in the application. Use to reverse the effect of disable connects . By default, connections are enabled.
disable connects	Prevent any user with a permission lower than Application Manager from making connections to the databases that require the databases to be started. This includes starting the databases or performing the ESSCMD SELECT command on the databases. Database connections remain disabled for all databases in the application, until the application setting is re-enabled by the administrator.
	By default, connections are enabled.
enable security	When security is disabled, Essbase ignores all security settings in the application and treats all users as Application Managers. By default, security is enabled.
disable security	When security is disabled, Essbase ignores all security settings in the application and treats all users as Application Managers. By default, security is enabled.
comment	Enter an application description (optional). The description can contain up to 80 characters.

Keyword	Description
clear logfile	Delete the application log located in the application directory. A new log is created for entries recording subsequent application activity.
add variable	<p>Create an application-level substitution variable by name, and optionally assign a string value for the variable to represent. You can assign or change the value later using set variable. A substitution variable acts as a global placeholder for information that changes regularly. Substitution variables may be referenced by calculations and report scripts.</p> <p>If substitution variables with the same name exist at server, application, and database levels, the order of precedence for the variables is as follows: a database level substitution variable supersedes an application level variable, which supersedes a server level variable.</p>
drop variable	Remove a substitution variable and its corresponding value from the application.
rename to	Rename the application. When you rename an application, the application and the application directory (<code>ARBORPATH\App\application_name</code>) are renamed.
sync user	Synchronize the named user's information on this Essbase application with the latest matching user information found on Shared Services. To issue this statement, you must be an Administrator, Application Manager, or Database Manager.
sync group	Synchronize the named group's information on this Essbase application with the latest matching group information found on Shared Services. To issue this statement, you must be an Administrator, Application Manager, or Database Manager.
sync all_users_groups	Synchronize all user and group information on this Essbase application with the latest user and group information found on Shared Services. To issue this statement, you must be an Administrator, Application Manager, or Database Manager.
reregister	<p>Re-establish this Essbase application as a Shared Services application. This statement reregisters the application with Shared Services, in the event that you have:</p> <ul style="list-style-type: none"> ● deleted the application from Shared Services but kept using it in Essbase. ● changed the Essbase Administration Server location, name, or port number. ● changed the Essbase Server name or port number. <p>To issue this statement, you must be an Administrator or Application Manager.</p>
all reregister	<p>Re-establish this and all other Essbase applications as Shared Services applications. This statement reregisters the applications with Shared Services, in the event that you have:</p> <ul style="list-style-type: none"> ● deleted the application from Shared Services but kept using it in Essbase. ● changed the Essbase Administration Server location, name, or port number. ● changed the Essbase Server name or port number. <p>To issue this statement, you must be an Administrator or Application Manager on all applications; for any applications for which you do not have sufficient permissions, the re-registration will be skipped with a warning.</p>

Example

```
alter application ASOSamp set cache_size 64MB;
```

Sets the maximum size of the aggregate storage cache to 64 MB.

```
alter application ASOSamp disable commands;
```

Prevents all users from making requests to the application scope. Use this statement before performing application-wide update and maintenance operations.

```
alter application ASOSamp comment 'Aggregate storage application';
```

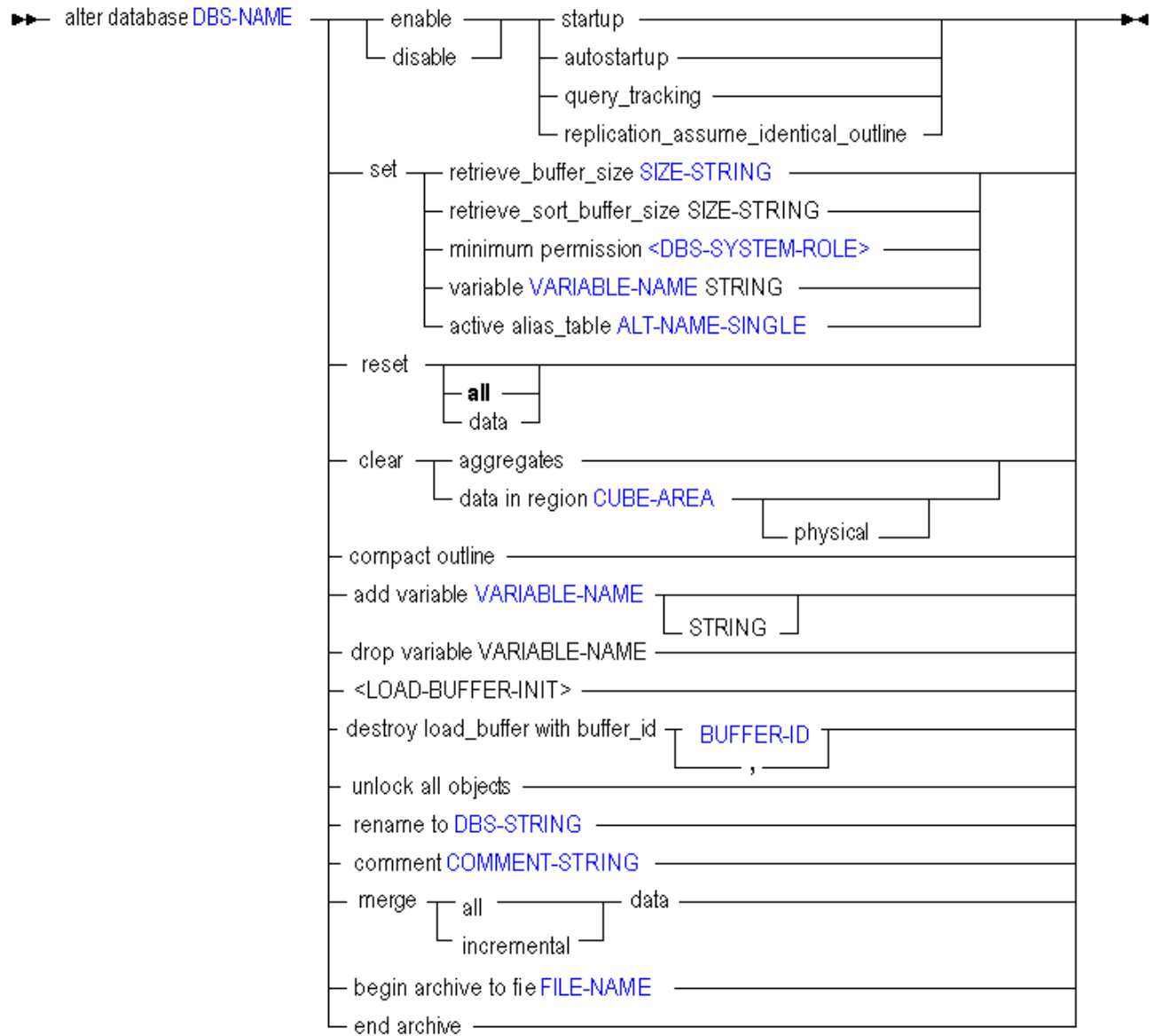
Attaches a descriptive comment to the ASOSamp application.

Alter Database (Aggregate Storage)

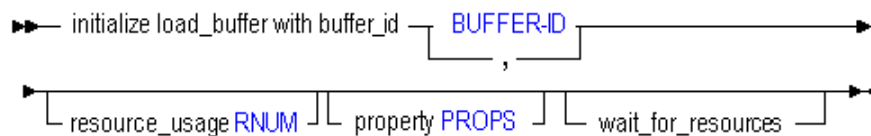
[Click here for non-aggregate storage version](#)

Change database-wide settings. Permission required: create_application.

Syntax



<LOAD-BUFFER-INIT> ::=



You can change the following database-wide settings using **alter database**.

Keyword

Description

enable startup	Enable users to start the database directly or as a result of requests requiring the database to be started. Startup is enabled by default.
disable startup	Prevent all users from starting the database directly or as a result of requests that would start the database. Startup is enabled by default.

Keyword	Description
enable autostartup	Automatically start the database when the application to which it belongs starts. Autostartup is enabled by default. This setting is applicable only when startup is enabled.
disable autostartup	Prevent automatic starting of the database when the application to which it belongs starts. Autostartup is enabled by default.
enable query_tracking	<p>Begin collecting query data for this database, to be used for query-based view optimization.</p> <p>To utilize the results of query tracking, use the optional based on query_data clause found in either of the following statements:</p> <ul style="list-style-type: none"> ● query database <dbs-name> list existing_views ● execute aggregate process ● execute aggregate selection <p>Query tracking is disabled by default.</p>
disable query_tracking	Stop collecting query data for query-based view optimization. Query tracking is disabled by default.
set retrieve_buffer_size	Change the database retrieval buffer size. This buffer holds extracted row data cells before they are evaluated by the RESTRICT or TOP/BOTTOM Report Writer commands. The default size is 10 KB. The minimum size is 2 KB. Increasing the size may improve retrieval performance.
set retrieve_sort_buffer_size	Change the database retrieval sort buffer size. This buffer holds data until it is sorted. The Report Writer and Essbase Query Designer use the retrieval sort buffer. The default size is 10 KB. The minimum size is 2 KB. Increasing the size may improve retrieval performance.
set minimum permission	Set a level of permission that all users or groups can have to the database. Users or groups with higher granted permissions than the minimum permission are not affected.
set variable	Change the value of an existing substitution variable on the database. The value must not exceed 256 bytes. It may contain any character except a leading ampersand (&).
set active alias_table	Set an alias table as the primary table for reporting and any additional alias requests. Only one alias table can be used at a time. This setting is user-specific; it only sets the active alias table for the user issuing the statement.
reset	<p>Clear all data and linked-reporting objects from the database, but preserve the outline.</p> <p>Note: If kernel queries are running when a clear data operation starts, the clear data operation waits for the kernel queries to complete and then the clear data operation proceeds. This information also applies to the reset all and reset data grammar.</p>
reset all	Clear all data, Linked Reporting Objects, and the outline.
reset data	Same as using reset .

Keyword	Description
clear aggregates	Delete all aggregate views.
compact outline	Compact the outline file to remove the records of members that have been deleted. Compaction helps keep the outline file at an optimal size.
add variable	<p>Create a database-level substitution variable by name, and optionally assign a string value for the variable to represent. You can assign or change the value later using <code>set variable</code>. A substitution variable acts as a global placeholder for information that changes regularly. Substitution variables may be referenced by calculations and report scripts.</p> <p>If substitution variables with the same name exist at server, application, and database levels, the order of precedence for the variables is as follows: a database level substitution variable supersedes an application level variable, which supersedes a server level variable.</p>
drop variable	Remove a substitution variable and its corresponding value from the database.
initialize load_buffer	<p>Create a temporary buffer in memory for loading data.</p> <p>Data load buffers are used in aggregate storage databases for allocations, custom calculations, and lock and send operations. Multiple data load buffers can exist on a single aggregate storage database.</p> <p>You can control the share of aggregate storage cache resources the load buffer is allowed to use and how long to wait for resources to become available before aborting load buffer operations. You can also set properties that determine how missing and zero values, duplicate values, and multiple values for the same cell in the data source are processed.</p> <ul style="list-style-type: none"> ● resource_usage ● property ● <code>wait_for_resources</code>: Waits up to the amount of time specified by the ASOLOADBUFFERWAIT configuration setting in <code>essbase.cfg</code> for resources to become available in order to process load buffer operations. The default value is 10 seconds.
destroy load_buffer	Destroy the temporary data-load memory buffer.
unlock all objects	Unlock all objects on the database that are in use by a user or process.
rename to	Rename the database. When you rename a database, the database directory is also renamed.
comment	Create a description of the database. The maximum number of characters is 80. This description is available to database administrators. To annotate the database for Spreadsheet Add-in users, use <code>set note</code> .
merge all incremental data [remove_zero_cells]	<p>Merge incremental data slices. Use these keywords:</p> <ul style="list-style-type: none"> ● <code>all</code>—Merge all incremental data slices into the main database slice. ● <code>incremental</code>—Merge all incremental data slices into a single data slice. The main database slice is not changed. ● (Optional) <code>remove_zero_cells</code>—When merging incremental data slices, remove cells that have a value of zero (logically clearing data from a region results in cell with a value of zero).

Keyword**Description**

clear data in region ...

Clear the data in the specified region.

There are two methods for clearing data from a region:

- Physical, in which the input cells in the specified region are physically removed from the aggregate storage database. The process for physically clearing data completes in a length of time that is proportional to the size of the input data, not the size of the data being cleared. Therefore, you might typically use this method only when you need to remove large slices of data.

Use the MaxL statement with the physical keyword:

```
alter database appname.dbname clear data in region  
'MDX set expression' physical;
```

- Logical, in which the input cells in the specified region are written to a new data slice with negative, compensating values that result in a value of zero for the cells you want to clear. The process for logically clearing data completes in a length of time that is proportional to the size of the data being cleared. Because compensating cells are created, this option increases the size of the database.

Use the MaxL statement without a keyword:

```
alter database appname.dbname clear data in region  
'MDX set expression';
```

The region must be symmetrical. Members in any dimension in the region must be stored members. When physically clearing data, members in the region can be upper-level members in alternate hierarchies. (If the region contains upper-level members from alternate hierarchies, you may experience a decrease in performance.) Members cannot be dynamic members (members with implicit or explicit MDX formulas), nor can they be from an attribute dimension.

To remove cells with a value of zero, use the **alter database** MaxL statement with the **merge** grammar and the **remove_zero_cells** keyword.

enable
replication_assume_identical_outline

Optimize the replication of an aggregate storage database when the aggregate storage database is the target and a block storage database is the source and the two outlines are identical.

Replication optimization affects only the target aggregate storage application; the source block storage application is not affected. This functionality does not apply to block storage replication.

This statement can be enabled only at the database level. To enable this functionality at the server or application (or database) level, use the [REPLICATIONASSUMEIDENTICALOUTLINE](#) configuration setting in the `essbase.cfg` file.

disable
replication_assume_identical_outline

Do not optimize the replication of an aggregate storage database when the aggregate storage database is the target and a block storage database is the source and the two outlines are identical.

Keyword	Description
begin archive to file	<p>Prepare the database for backup by an archiving program, and prevent writing to the files during backup.</p> <p>Begin archive achieves the following outcomes:</p> <ul style="list-style-type: none"> • Switches the database to read-only mode. The read-only state persists, even after the application is restarted, until it is changed back to read-write using <code>end archive</code>. • Creates a file containing a list of files that need to be backed up. Unless a different path is specified, the file is stored in the database directory. <p>Begin archive and end archive do not perform the backup; they simply protect the database during the backup process.</p>
end archive	Return the database to read-write mode after backing up the database files.

Example

```
alter database AsoSamp.Sample clear aggregates;
```

Deletes all aggregate views in the AsoSamp.Sample database.

```
alter database AsoSamp.Sample initialize load_buffer with buffer_id 1;
```

See [“Loading Data Using Buffers” on page 921](#).

```
alter database AsoSamp.Sample initialize load_buffer with buffer_id 1 resource_usage .5
property ignore_missing_values, ignore_zero_values;
```

Creates a data-load buffer in memory for the AsoSamp.Sample database. The buffer can use only 50% of available resources. Missing values and zeros in the data source are ignored.

```
alter database AsoSamp.Sample disable query_tracking;
```

Turns off the harvesting of query data for the AsoSamp.Sample database.

```
alter database AsoSamp.Sample merge all data;
```

Merges all incremental data slices into the main slice in the AsoSamp.Sample database.

```
alter database AsoSamp.Sample merge incremental data;
```

Merges all incremental data slices into a single data slice within the AsoSamp.Sample database.

```
alter database AsoSamp.Sample merge all data remove_zero_cells;
```

Merges all incremental data slices into the main slice in the AsoSamp.Sample database, and removes cells with a value of zero.

```
alter database AsoSamp.Sample clear data in region '{Jan, Budget}';
```

Clears all Budget data for the month of Jan, using the logical method, from the AsoSamp.Sample database.

```
alter database AsoSamp.Sample clear data in region '{Jan, Budget}' physical;
```

Clears all Budget data for the month of Jan, using the physical method, from the AsoSamp.Sample database.

```
alter database AsoSamp.Sample clear data in region 'CrossJoin({Jan},{Forecast1,Forecast2})';
```

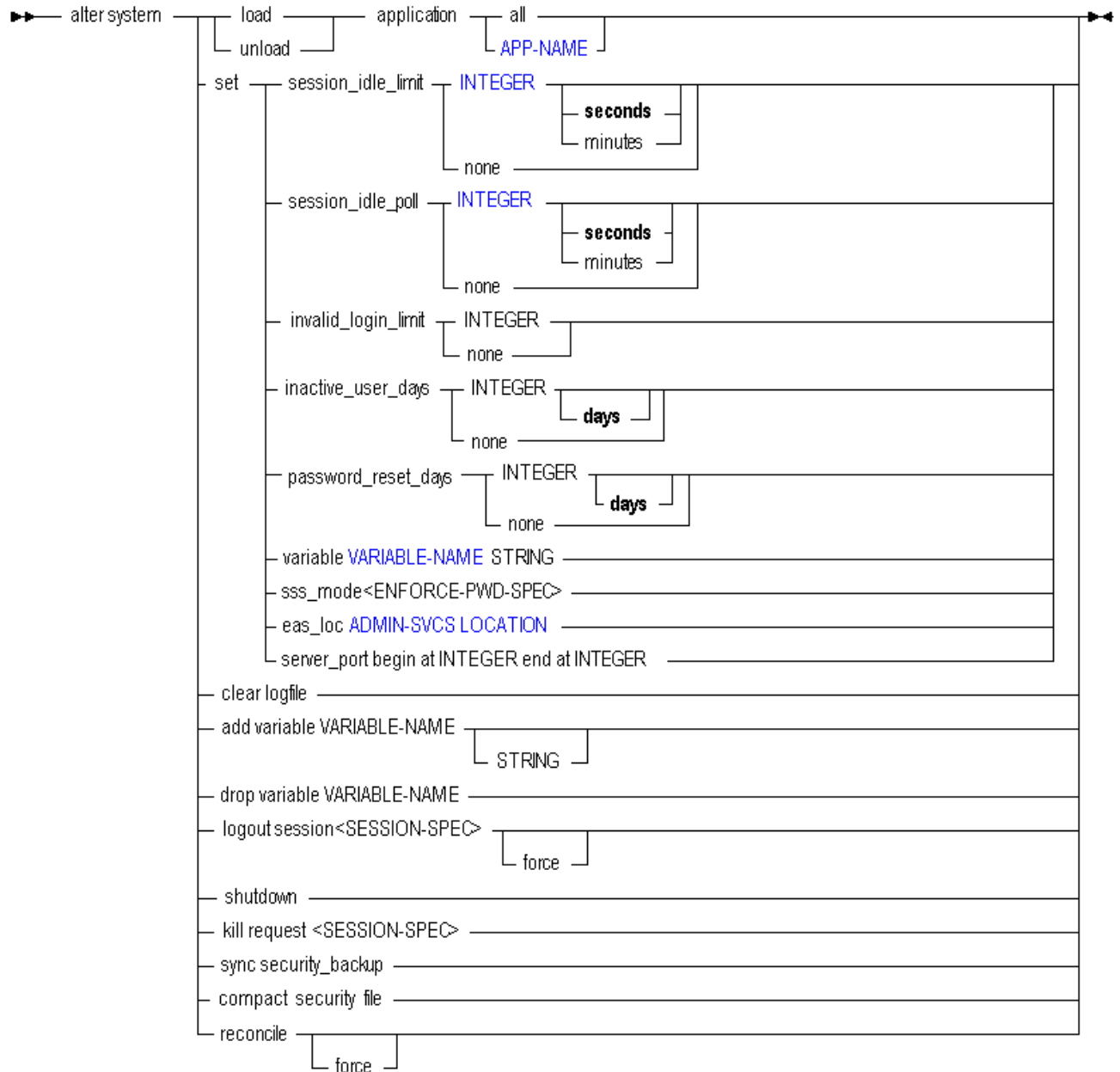
Clears all January data for the Forecast1 and Forecast2 scenarios from the AsoSamp.Sample database.

Alter System (Aggregate Storage)

[Click here for non-aggregate storage version](#)

Change the state of the Essbase Server. Start and stop applications, delete application log files, manipulate system-wide variables, manage password and login activity, disconnect users, kill processes, back up the security file, and shut down the server. Permission required: Administrator.

Syntax



You can change the following system-wide settings using `alter system`.

Keyword	Description
load application	Start an application, or start all applications on the Essbase Server.
unload application	Stop an application, or stop all applications on the Essbase Server.

Keyword	Description
set session_idle_limit	<p>Set the interval of time permitted for a session to be inactive before Essbase Server logs off the user. The minimum limit that you can set is five minutes (or 300 seconds). When the session idle limit is set to none, all users can stay logged on until the Essbase Server is shut down.</p> <p>The default user idle logout time is 60 minutes. When a user initiates a calculation in the background, after 60 minutes the user is considered idle and is logged out, but the calculation continues in the background.</p> <p>Because the user may mistakenly assume that the calculation stopped because he or she was logged out, you can do one of the following to correct the user experience:</p> <ul style="list-style-type: none"> ● Run the calculation in the foreground ● Increase the session idle limit in to a time that exceeds the duration of the calculation, or to none
set session_idle_poll	<p>Set the time interval for inactivity checking and security-backup refreshing. The time interval specified in the session idle poll gives Essbase instructions:</p> <ul style="list-style-type: none"> ● Tells it how often to check whether user sessions have passed the allowed inactivity interval indicated by <code>session_idle_limit</code> in the alter system statement. ● Tells it how often to refresh the security backup file. If <code>session_idle_poll</code> is set to zero, the security backup file is still refreshed every five minutes.
set invalid_login_limit	<p>Set the number of unsuccessful login attempts allowed by any user before the user account becomes disabled. When you change this setting, the counter resets to 0. When the invalid login limit is set to none, there is no limit. By default, there is no limit.</p>
set inactive_user_days	<p>Set the number of days a user account may remain inactive before the system disables it. The counter resets when the user logs in, is edited, or is activated by an administrator. When the inactive days limit is set to none, user accounts remain enabled even if they are not used. By default, there is no limit.</p>
set password_reset_days	<p>Set the number of days users may retain passwords. After the allotted number of days, users are prompted at login to change their passwords. The counter resets for a user when the user changes the password, is edited, or is activated by an administrator. When the password reset days limit is set to none, there is no built-in limit for password retention. By default, there is no limit.</p>
set variable	<p>Change the value of an existing substitution variable on the system. The value must not exceed 256 bytes. It may contain any character except a leading ampersand (&).</p>

Keyword	Description
set sss_mode	<p>Migrate Essbase Server and any existing users and groups to Shared Services security mode. Minimum permission required: Administrator. After you have converted to Shared Services security mode, you cannot revert to native security mode.</p> <p>Password Enforcement Grammar:</p> <ul style="list-style-type: none"> ● enforce username_as_password—Create passwords that are the same as user names for users being migrated to Shared Services. <p>Note: The passwords are created in lowercase letters, even if the user name includes uppercase letters. For example, if a user name <code>KSmith</code> is migrated with this option, the password will be <code>ksmith</code>.</p> <ul style="list-style-type: none"> ● enforce auto_password—Automatically generate new passwords for users being migrated to Shared Services. To see the generated passwords, use <code>display user all in shared_services_native with auto_password;</code> <p>Optionally save the generated passwords to a nondefault file location. If specifying a file name that already exists, use the <code>force</code> keyword to overwrite the file.</p> <p>If file name and location are not specified, passwords are saved by default to <code>\$ARBORPATH\bin\MigratedUsersPassword.txt</code>.</p> <ul style="list-style-type: none"> ● enforce password <PASSWORD>—Generate the specified password for users being migrated to Shared Services.
set eas_loc	Set or change the Essbase Administration Server location that will be registered with Shared Services upon application creation or migration.
set server_port	<p>Expand a port range specified in <code>essbase.cfg</code>. Each Essbase application uses two ports from this range. If no more ports are available, an error message is displayed.</p> <p>Note: You can expand port ranges only so that the beginning port range is less than <code>SERVERPORTBEGIN</code> and the ending port range is greater than <code>SERVERPORTEND</code>.</p>
clear logfile	Clear accumulated entries from the Essbase Server log located in the <code>Essbase</code> directory. New log entries are created to record subsequent activity.
add variable	<p>Create a system-level substitution variable by name, and optionally assign a string value for the variable to represent. You can assign or change the value later using <code>set variable</code>. A substitution variable acts as a global placeholder for information that changes regularly. Substitution variables may be referenced by calculations and report scripts.</p> <p>If substitution variables with the same name exist at server, application, and database levels, the order of precedence for the variables is as follows: a database-level substitution variable supersedes an application-level variable, which supersedes a server-level variable.</p>
drop variable	Remove a substitution variable and its corresponding value from the system.
logout session all	Terminate all user sessions currently running on the Essbase Server.
logout session...force	Terminate a session (or sessions) even if it is currently processing a request. The request is allowed to proceed to a safe point, and then the transaction is rolled back.
logout session <session-id>	Terminate a session by its unique session ID number. To see the session ID number, use display session .
logout session by user	Terminate all current sessions by a particular user, either across the entire Essbase Server, or limited to a specific application or database.

Keyword	Description
logout session by user on application	Terminate all current sessions by a particular user across a specific application.
logout session by user on database	Terminate all current sessions by a particular user across a specific database.
logout session on application	Terminate all current user sessions across a specific application.
logout session on database	Terminate all current user sessions across a specific database.
shutdown	Shut down the Essbase Server.
kill request all	Terminate all current requests on the Essbase Server.
kill request <session-id>	Terminate the current request indicated by the session ID. You can obtain session IDs using display session .
kill request by user	Terminate all current requests by the specified user on the Essbase Server.
kill request on application	Terminate all current requests on the specified application.
kill request on database	Terminate all current requests on the specified database.
sync security_backup	<p>Check whether the security backup file is the same as the security file, and if not, synchronize the security backup file with the current state of Essbase security. The effect is to refresh the backup file with any additions, changes, or deletions related to applications, databases, users, groups, filters, permissions, substitution variables, locked objects, and system settings.</p> <p>If <code>sync security_backup</code> is not issued directly as described above, the security backup file is checked/refreshed automatically at the same frequency with which session inactivity is checked globally. The default inactivity check interval is every five minutes. To change the interval, use <code>set session_idle_poll</code>, or see the <i>Oracle Essbase Administration Services Online Help</i>.</p>
compact security file	Defragment the security file. Fragmentation can gradually develop when objects such as users, groups, applications, or databases are removed or changed. Please note that this operation slows down agent activity until the operation is completed, which could take a few minutes.

Keyword	Description
reconcile	<p>When Essbase is started using a security backup file (<i>essbase_timestamp.bak</i>) instead of <i>essbase.sec</i>, reconcile the security file to match the state of Essbase on an external disk. This grammar displays discrepancies in application and database information between the security file and the external disk:</p> <ul style="list-style-type: none"> ● If an application folder is on the disk but not in the security file, display a message indicating the discrepancy. (Essbase checks for the presence of a <i>appname/appname.app</i> file in the <i>ARBORPATH/app</i> directory.) The <i>force</i> option does not apply in this scenario. ● If an application file is in the security file but not on the disk, display a message indicating the discrepancy. The <i>force</i> option removes the application from the security file. ● If an application database folder is on the disk but not in the security file, display a message indicating the discrepancy. (Essbase checks for the presence of a <i>dbname/dbname.otl</i> file in the <i>ARBORPATH/app/appname</i> directory.) The <i>force</i> option does not apply in this scenario. ● If an application database file is in the security file but not on the disk, display a message indicating the discrepancy. The <i>force</i> option removes the database from the security file.

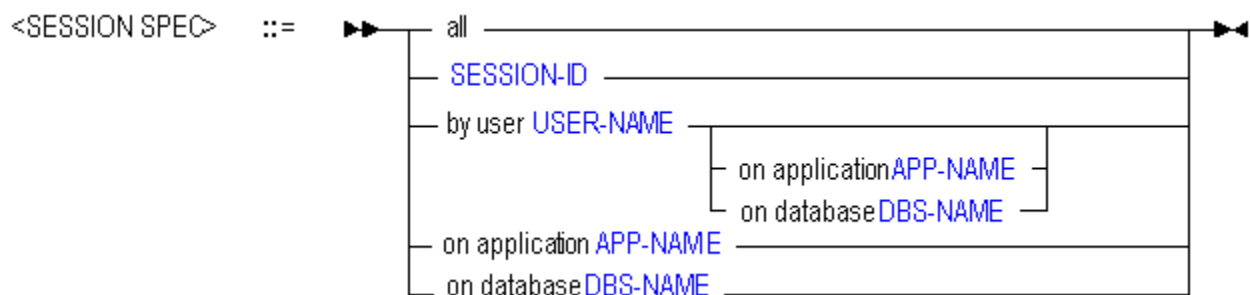
Notes

SESSION SPECIFICATION

A *session* is a single user connection to Essbase Server. The session can be identified by keywords and names indicating context, or by a unique session ID number.

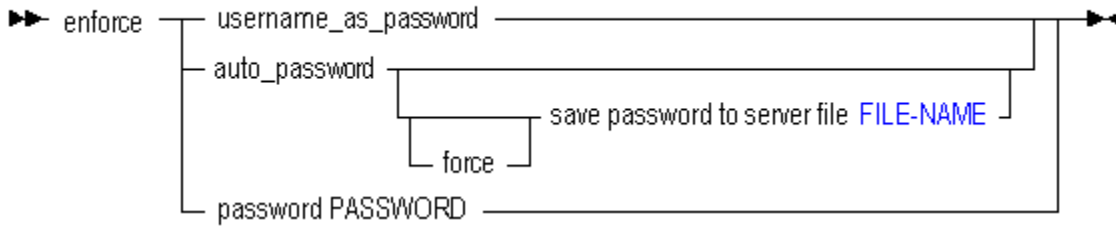
A *request* is a query sent to Essbase Server by a user or by another process; for example, starting an application or restructuring a database outline. Only one request at a time can be processed in each session.

If a session is processing a request at the time that an administrator attempts to terminate the session, the administrator must either terminate the request first, or use the *force* keyword available with *alter system* to terminate the session *and* the current request.



PASSWORD ENFORCEMENT SPECIFICATION

<ENFORCE-PWD-SPEC> ::=



Example

```
alter system unload application Sample;
```

Stops the Sample application, if it is currently running.

```
alter system logout session by user Fiona;
```

Disconnects Fiona from any applications or databases to which she is connected.

Note: To log out a user, log out the sessions owned by that user.

```
alter system set password_reset_days 10;
```

Specifies that all users will be prompted after 10 days to change their passwords. The day count for any user is reset when the user changes the password or is edited or reactivated by an administrator.

```
alter system set sss_mode enforce password "password";
```

Migrates the Essbase Server to Shared Services security mode, specifying the initial password for all users.

Create Application (Aggregate Storage)

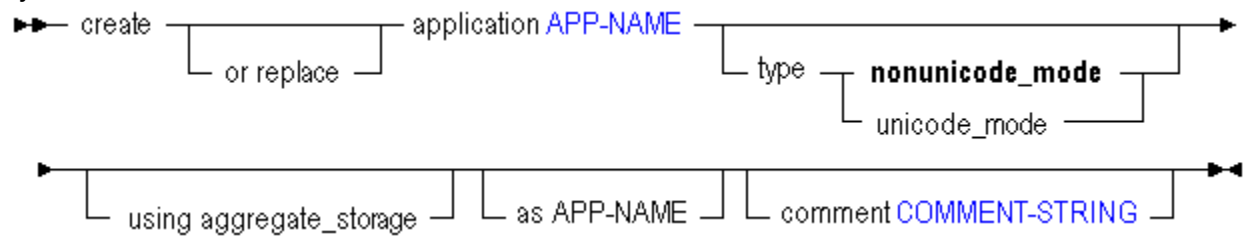
[Click here for non-aggregate storage version](#)

Create or re-create an application, either from scratch or as a copy of another application on the same system. APP-NAME must consist of 8 or fewer characters. Avoid spaces and special characters when naming applications and databases. Application names are not case-sensitive.

Permissions required: Essbase create_application role and Oracle's Hyperion® Shared Services Project Manager role.

To copy an application, Application Manager permission on the source application is also required.

Syntax



You can create an application in the following ways using the aggregate storage version of `create application`.

Keyword	Description
<code>create application</code>	Create a new application. Application names are not case-sensitive.
<code>create or replace application</code>	Create an application, or replace an existing application of the same name. Application names are not case-sensitive.
<code>...type nonunicode_mode</code>	Create a Non Unicode-mode application. This is also the default if these keywords are omitted.
<code>...type unicode_mode</code>	Create a Unicode-mode application.
<code>...using aggregate_storage</code>	Create an application using an aggregate storage model. Only one database per application is allowed. Selecting to use aggregate storage model for an application is non-reversible. Use the aggregate storage model if the following is true for your database: <ul style="list-style-type: none">• The database is sparse and has many dimensions, or a large hierarchical depth of members in the dimensions.• The database is used primarily for read-only purposes; there are few or no data updates.• There are no formulas on the outline except in the dimension tagged as Accounts.• Calculation of the database is frequent and highly aggregational, with no dependency on calculation scripts.
<code>create application as</code>	Create an application as a copy of another application. Application names are not case-sensitive. You cannot copy block storage applications to aggregate storage applications or vice versa. The copy will always use the same storage as the original. However, you can convert an outline from a block storage database to an aggregate storage database, using create outline .
<code>comment</code>	Create an application description (optional). The description can contain up to 80 characters.

Example

```
create application Sample2 using aggregate_storage comment 'aggregate storage application.';
```

Creates a new aggregate storage application called Sample2, with an associated comment.

Create Database (Aggregate Storage)

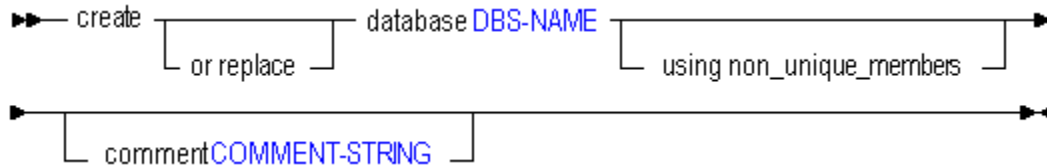
[Click here for non-aggregate storage version](#)

Create or re-create a database for an aggregate storage application.

The syntax for creating an aggregate storage database is the same as for creating a block storage database, except that the currency database option is not supported. You must create an aggregate storage database as part of an aggregate storage application.

Permission required: Application Manager.

Syntax



Use **create database** to create a database in the following ways:

Keyword	Description
create database	Create a new database. Database names are not case-sensitive.
create or replace database	Create a database, or replace an existing database of the same name. Database names are not case-sensitive.
create database using non_unique_members	Create a database that supports the use of duplicate member names. Once you have created a database with a duplicate member outline, you cannot convert it back to a unique member outline. For more information about duplicate member names, see the <i>Oracle Essbase Database Administrator's Guide</i> chapter titled "Creating and Working With Duplicate Member Outlines."
comment	Create a database description (optional). The description can contain up to 80 characters.

Notes

- You cannot create an aggregate storage database as a copy of another aggregate storage database. Only one aggregate storage database is allowed per application.
- You cannot copy a block storage database to an aggregate storage database. For an example of how to create an aggregate storage application and database based on a block storage application and database, see [“Creating an Aggregate Storage Sample Using MaxL” on page 920](#).

Example

```
create or replace database Sample.Basic comment 'This is a test.';
```

Creates a database called Basic within the Sample application. If a database named Basic within the Sample application already exists, it is overwritten.

Create Outline (Aggregate Storage)

Create an aggregate storage outline based on a block storage outline. The outline you are creating must be for an aggregate storage database that is local to your current login session. The block-storage database you are using as a source can be remote. If a remote host is specified, you can

also specify a user name and password if the connection is remote. Permission required: Database Manager.

Essbase supports the following scenarios for converting block storage outlines to aggregate storage outlines:

- Non-Unicode block storage outline to non-Unicode aggregate storage outline
- Non-Unicode block storage outline to Unicode aggregate storage outline
- Unicode block storage outline to Unicode aggregate storage outline

The following conversion scenarios are not supported:

- Unicode block storage outline to non-Unicode aggregate storage outline
- Aggregate storage outline to a block storage outline

Syntax

```
create [ or replace ] outline on aggregate_storage database DBS-NAME as outline
on database DBS-NAME [ at HOST-NAME ] [ as USER-NAME identified by PASSWORD ]
```

You can create an outline in the following ways using **create outline**.

Keyword	Description
create outline...	Create an aggregate-storage database outline based on a block storage outline. If an outline of the same name already exists, it is replaced.
create or replace outline...	This statement has the same result as <code>create outline</code> above.
at HOST-NAME	If the block-storage database you are using as a source is remote, specify the host name.
as USER-NAME identified by PASSWORD	If the block-storage database you are using as a source is remote, specify the host name. If the connection is also remote (requires a different authentication), provide the user name and password, as you would do when creating a remote partition.

Example

```
create or replace outline on aggregate_storage database Sample2.Basic2 as outline on
database sample.basic;
```

Creates an aggregate storage outline based on the Sample Basic outline. For a complete example of how to create an aggregate storage version of a block storage database, see [“Creating an Aggregate Storage Sample Using MaxL” on page 920](#).

Display Tablespace

View details about a tablespace. Tablespaces are applicable only to aggregate storage databases. Permission required: Application Manager. This statement requires the application to be started.

Syntax

►►— display tablespace **TABLSP-NAME** —►◄

Example

```
set column_width 50; /* so file_location will not be truncated */
display tablespace ASOSamp.'default';
```

This example displays the following output:

Column Header	Contents
file_location	C:\Hyperion\products\Essbase\EssbaseServer\APP\
max_file_size	56
max_disk_size	4294967295

Execute Allocation

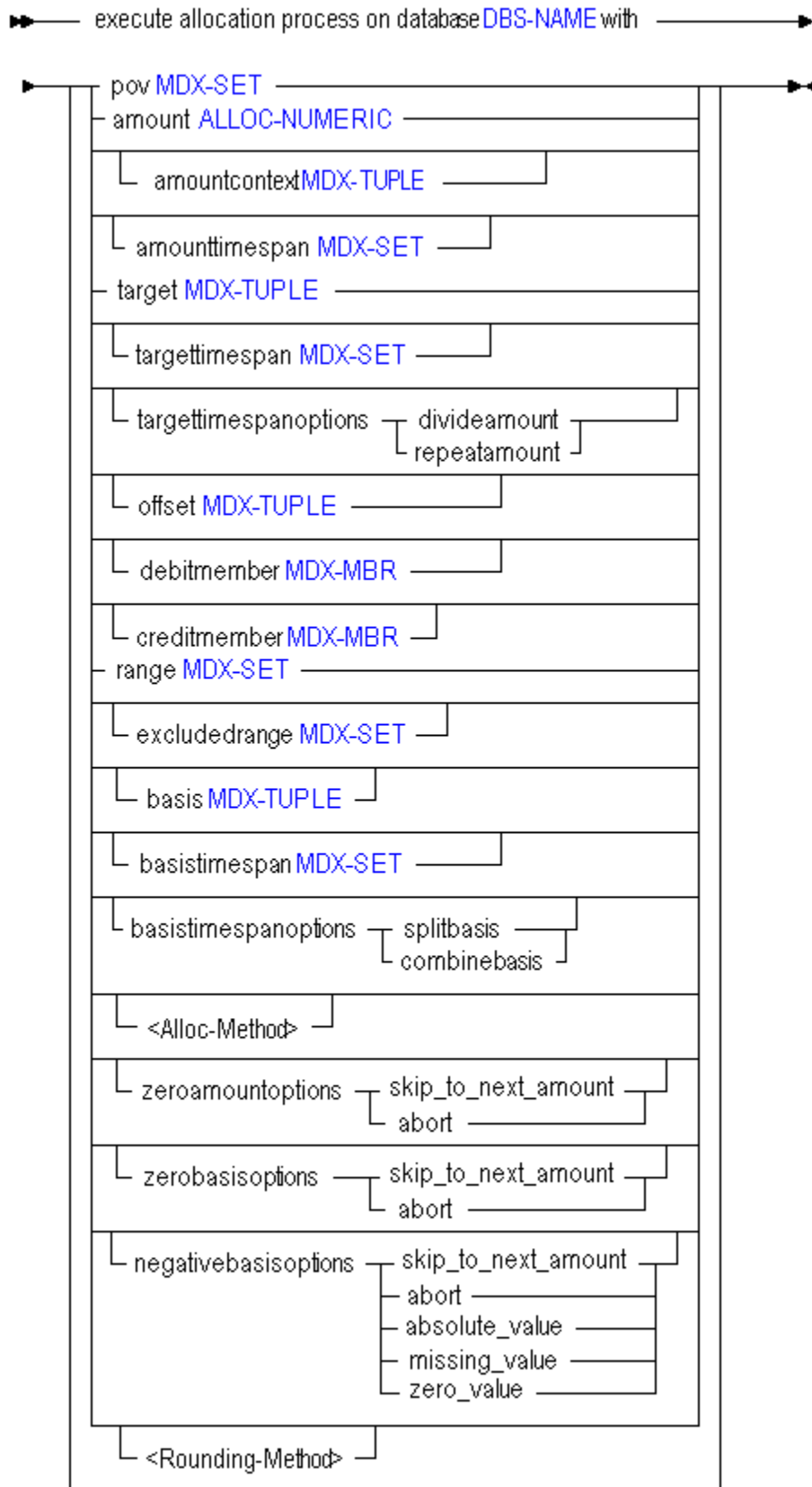
Allocate one or more given source amounts to a target range of cells in an aggregate storage database. The source amount can be allocated to the target proportionately to a given basis, or the source amount can be spread evenly to the target region.

Allocations are typically used in the budgeting process to distribute revenues or costs.

Minimum permission required: execute.

For more information about allocations and to understand the input parameters, see “Performing Custom Calculations and Allocations on Aggregate Storage Databases” in the *Oracle Essbase Database Administrator's Guide*.

Syntax



Keyword	Description
pov <mdx-set>	Required. Provide an MDX set defining the context region in which the allocation is performed.
amount <alloc-numeric>	Required. Provide an MDX numeric value expression indicating the amount to be allocated.
amountcontext <mdx-tuple>	Optional. Provide an MDX tuple with one member from each dimension missing from pov and amount . This clause is required when amount is an arithmetic expression and pov does not specify two or more dimensions. It should not be used otherwise.
amounttimespan <mdx-set>	Optional. Provide an MDX set indicating one or more time periods to be considered for the amount. The amount value is aggregated over the specified time periods, and the aggregated amount value is allocated. Time periods must be level 0 members in a Time dimension.
target <mdx-tuple>	Required. Provide an MDX tuple defining the database region where results are written.
targettimespan <mdx-set>	Optional. Provide an MDX set indicating one or more time periods to be considered for the target. Time periods must be level 0 members in a Time dimension.
targettimespanoptions	Optional, but required if targettimespan is used. Select a method for allocating values across the target time span: <ul style="list-style-type: none"> ● divideamount—Divide the amount evenly across the time periods ● repeatamount—Repeat the amount across the time periods
offset <mdx-tuple>	Optional. If offsetting entries are used, provide an MDX tuple defining the location in the database where an offsetting value is written for each source amount.
debitmember <mdx-mbr>	Optional. If double-entry accounting is used, provide an MDX member expression indicating the member to which positive result values are written.
creditmember <mdx-mbr>	Optional. If double-entry accounting is used, provide an MDX member expression indicating the member to which negative result values are written.
range <mdx-set>	Required. Provide an MDX set indicating the database region in which allocated values are calculated and written.
excludedrange <mdx-set>	Optional. Provide an MDX set specifying locations in the range where you do not want allocation values written.
basis <mdx-tuple>	Required in most cases. Provide an MDX tuple that, when combined with the range, defines the location of basis values that determine how the amount is allocated. The basis can consist of upper-level or level 0 members. Optional if the allocation method used is spread , and no values are skipped; required otherwise. Basis must be omitted when the allocation method spread is used without skip options.
basistimespan <mdx-set>	Optional. Provide an MDX set that indicates one or more time periods to be considered for the basis. Time periods must be level 0 members in a Time dimension.
basistimespanoptions	Optional, but required if basistimespan is used. Select a method for using the basis time span: <ul style="list-style-type: none"> ● splitbasis—Use the basis value for each time period individually ● combinebasis—Use the sum of the basis values across the time periods specified by basistimespan

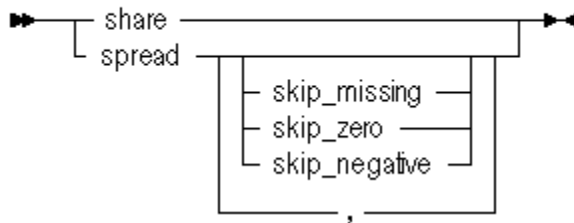
Keyword	Description
share	Optional. Specify to allocate the amount(s) proportionately to the basis values. For syntax, see Allocation Method Specification in Notes.
spread	Optional. Specify to allocate the amount(s) evenly. For syntax, see Allocation Method Specification in Notes. You can include one or more of the following skip options when using spread allocation: <ul style="list-style-type: none"> ● skip_missing–Skip missing basis values ● skip_zero–Skip zero basis values ● skip_negative–Skip negative basis values
zeroamountoptions	Optional. If omitted, zero or #MISSING amount values are allocated. Otherwise, specify treatment of amount values that are zero or #MISSING: <ul style="list-style-type: none"> ● skip_to_next_amount–Skip to the next nonzero, non-#MISSING amount value ● abort–Cancel the entire allocation operation
zerobasisoptions	Optional. For share , this option specifies the action when the sum of all basis values is zero. For spread , this option specifies the action when all the basis values are skipped. Select one of the following options: <ul style="list-style-type: none"> ● skip_to_next_amount–Skip to the next nonzero, non-#MISSING amount value ● abort–Cancel the entire allocation operation
round	Optional. Specify rounding options. The following options are available: For syntax, see Rounding Method Specification in Notes. <ul style="list-style-type: none"> ● Round to a specified number of decimal places, using an integer or MDX numeric value expression. The value must be between 100 and -100, and is truncated if it is not a whole number. ● Perform rounding, but discard rounding errors ● Add rounding errors to the highest allocated value ● Add rounding errors to the lowest allocated value ● Provide an MDX tuple indicating a cell to which the rounding error should be added

Notes

- The clauses following the **with** keyword can be entered in any order, each separated by white space.
- Each clause can only be entered once.
- The **pov**, **amount**, **target**, **range**, and **basis** clauses are mandatory; the others are optional.
- You can specify only stored, level-0 members in all of the clauses except for **amount**, **amountcontext**, **basis**, and the number of rounding digits; for all other arguments, do not use upper-level members, attribute members, or dynamic calc members.

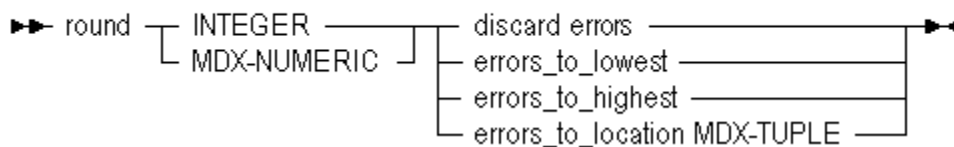
Allocation Method Specification

<Alloc-Method> ::=



Rounding Method Specification

<Rounding-Method> ::=



Example

The following statement executes an allocation. For a more complete use case, see “Performing Custom Calculations and Allocations on Aggregate Storage Databases” in the *Oracle Essbase Database Administrator's Guide*.

```
execute allocation on database glrpt.db with
pov          "Crossjoin({[VisionUS]},
              Crossjoin({[5740]},
              Crossjoin({[USD]},
              Descendants([Geography],[Geography].Levels(0)))))"
amount       "Jan + Feb"
amountcontext "[100], [Beginning Balance], [Actual], [CostCenter1]"
target       "([Allocation], [CostCenter1])"
offset       "([Allocation], [CostCenter1], [100], [YearNA])"
debitmember  "[Debit]"
creditmember "[Credit]"
range        "Crossjoin(Descendants([999], [Department].Levels(0)),
              Descendants([Year], [Year].Levels(0)))"
excludedrange "{[9994], [9995], [9996]}"
basis        "([SQFT], [Balance], [Actual], [CostCenter2])"
share
zeroamountoptions  abort
zerobasisoptions  abort
negativebasisoptions  zero_value
targettimespanoptions  divideamount
round              "Currency.CurrentMember.CurrencyPrecision"
errors_to_location "([101], [Jan])" ;
```

Execute Calculation (Aggregate Storage)

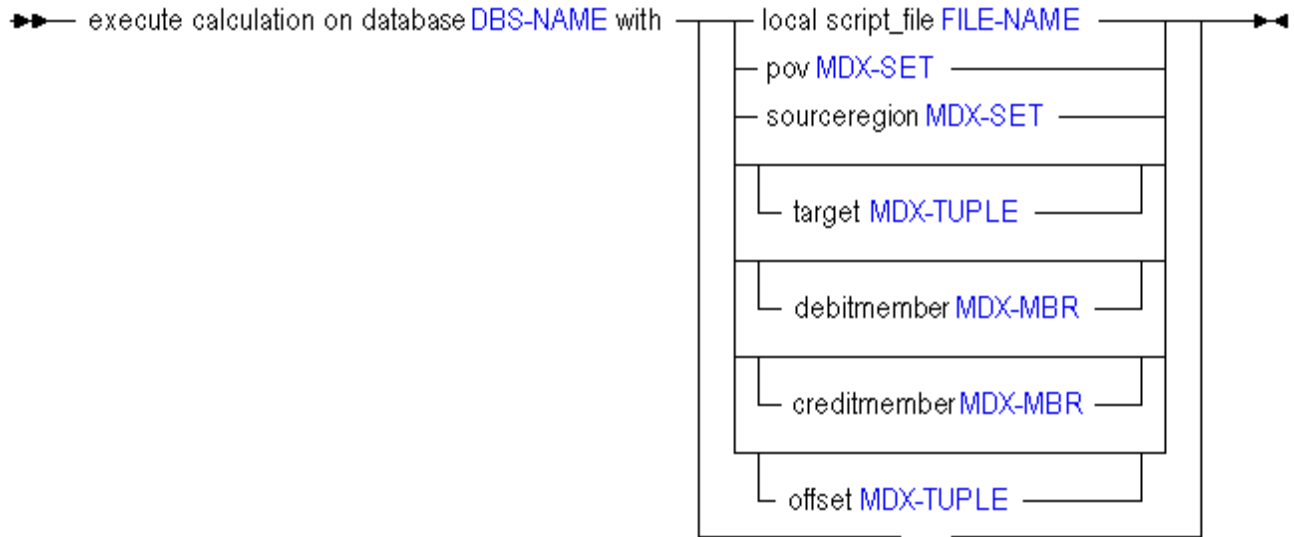
[Click here for non-aggregate storage version](#)

Execute a custom calculation script expressed in MDX, specifying the script file, source region, and point of view (POV). Optionally specify the target, offset, and debit or credit members.

Minimum permission required: execute.

For more information about custom calculation script parameters, see “Performing Custom Calculations and Allocations on Aggregate Storage Databases” in the *Oracle Essbase Database Administrator's Guide*.

Syntax



You can execute custom calculations with the following options:

Keyword	Description
local script_file	Required. Run the specified local calculation script file. Custom calculation scripts are expressed in MDX. The following is an example of a custom calculation script, <code>script.txt</code> . <pre>(AccountA, Proj1) := 100; ([AccountB], [Proj1]) := ([AccountB], [Proj1]) * 1.1; (AccountC, Proj1) := ((AccountB, Proj1, 2007) + (AccountB, Proj1)) / 2; (AccountA, Proj2) := ((AccountD, Proj1) + (AccountB, Proj2)) / 2;</pre> For information about writing custom calculation scripts, see “Performing Custom Calculations and Allocations on Aggregate Storage Databases” in the <i>Oracle Essbase Database Administrator's Guide</i> .
pov <mdx-set>	Required. Provide an MDX set defining the context region in which the calculation is performed. The calculation script will be executed once for every cross-product in the POV region.
sourceregion <mdx-set>	Required. Provide an MDX set specifying the region of the cube referred to by the formulas in the script. At a minimum, the source region should include all members from the right-hand sides of the assignment statements in the custom calculation script.
target <mdx-tuple>	Optional. Provide an MDX tuple defining the database region where results are written. You can use only stored, level-0 members in the tuple; do not use upper-level members, attribute members, or dynamic calc members.

Keyword	Description
debitmember <mdx-mbr>	Optional. If double-entry accounting is used, provide an MDX member expression indicating the member to which positive result values are written. You can specify only stored, level-0 members; do not use upper-level members, attribute members, or dynamic calc members.
creditmember <mdx-mbr>	Optional. If double-entry accounting is used, provide an MDX member expression indicating the member to which negative result values are written. You can specify only stored, level-0 members; do not use upper-level members, attribute members, or dynamic calc members.
offset <mdx-tuple>	Optional. If offsetting entries are used, provide an MDX tuple defining the location in the database where an offsetting value for each source amount is written. You can use only stored, level-0 members in the tuple; do not use upper-level members, attribute members, or dynamic calc members.

Notes

- The clauses following the **with** keyword can be entered in any order, each separated by white space.
- Each clause can only be entered once.
- The **script_file**, **pov**, and **sourceregion** clauses are mandatory; the others are optional.
- You can specify only stored, level-0 members on the left side of the assignment statement in the custom calculation script; do not use upper-level members, attribute members, or dynamic calc members.
- You can specify only stored, level-0 members in the following clauses: DebitMember, CreditMember, Target, and Offset.

Example

The following statement executes `script.txt` referenced above. For a sample use case, see “Performing Custom Calculations and Allocations on Aggregate Storage Databases” in the *Oracle Essbase Database Administrator's Guide*.

```
execute calculation on app.db with
  local script_file "script.txt"
  POV              "Crossjoin({[VisionUS]},
                    Crossjoin({[101]},
                              Crossjoin ({[Jan]},
                                        Crossjoin({[Scenario]},
                                                  Descendants(Geography, Geography.Levels(0))))))"
  Target           "(Allocation)"
  DebitMember      "[BeginningBalance_Debit]"
  CreditMember     "[BeginningBalance_Credit]"
  Offset           "([Account_000], [Project_000])"
  SourceRegion     "Crossjoin({[AccountB], [AccountD]},
                    Crossjoin({[Proj1], [Proj2]}, {[2007]}))" ;
```

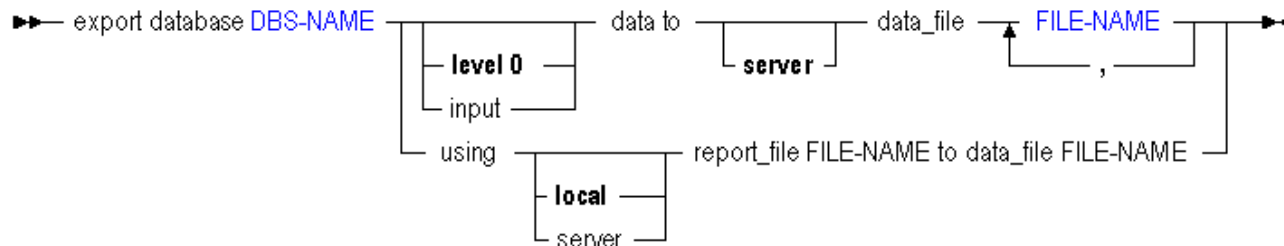
Export Data (Aggregate Storage)

[Click here for non-aggregate storage version](#)

Export level-0 data, which does not include calculated values, from an aggregate storage database. Export files are stored in the ARBORPATH/app directory on the server unless an absolute path is specified. To use Report Writer, export the data using a report file.

Minimum permission required: Read. This statement requires the database to be started.

Syntax



On aggregate storage databases, use **export data** to export in the following ways:

Keyword	Description
export database <db-name> level0 data...	Export level-0 input data to a text file. You cannot export aggregates, upper level data, or data from dynamically calculated members. Note: Exporting data does not clear the data from the database.
export database <db-name> input data...	This statement performs the same action as export database <db-name> level0 data...
export database <db-name> ...using...report_file...	Run a stored report script, exporting a subset of the database.

Notes

Exports on aggregate storage databases are limited as follows:

- You can export level-0 data only (level-0 data is the same as input data in aggregate storage databases).
- You cannot perform upper-level data export on an aggregate storage database.
- You cannot perform columnar export on an aggregate storage database.
- To export data in parallel, specify a comma-separated list of export files. The number of threads Essbase uses depends on the number of file names you specify. For parallel export on a very small database, it is possible that only a single file will be created, even though parallel export to multiple files is requested. In this case, the export file name will be the first file name given as input.
- During a data export, the export process allows users to connect and perform read-only operations.
- If the data for a thread exceeds 2 GB, Essbase may divide the export data into multiple files with numbers appended to the file names.

The naming convention for additional export files is as follows: `_1`, `_2`, etc. are appended to the additional file names. If the specified output file name contains a period, the numbers are appended before the period. Otherwise, they are appended at the end of the file name.

For example, if the given file name is `/home/exportfile.txt`, the next additional file is `/home/exportfile_1.txt`. If the file name is `/home/exportfile`, the next additional file is `/home/exportfile_1`.

Example

```
export database ASOSamp.Sample data to data_file 'exportfile.exp';
```

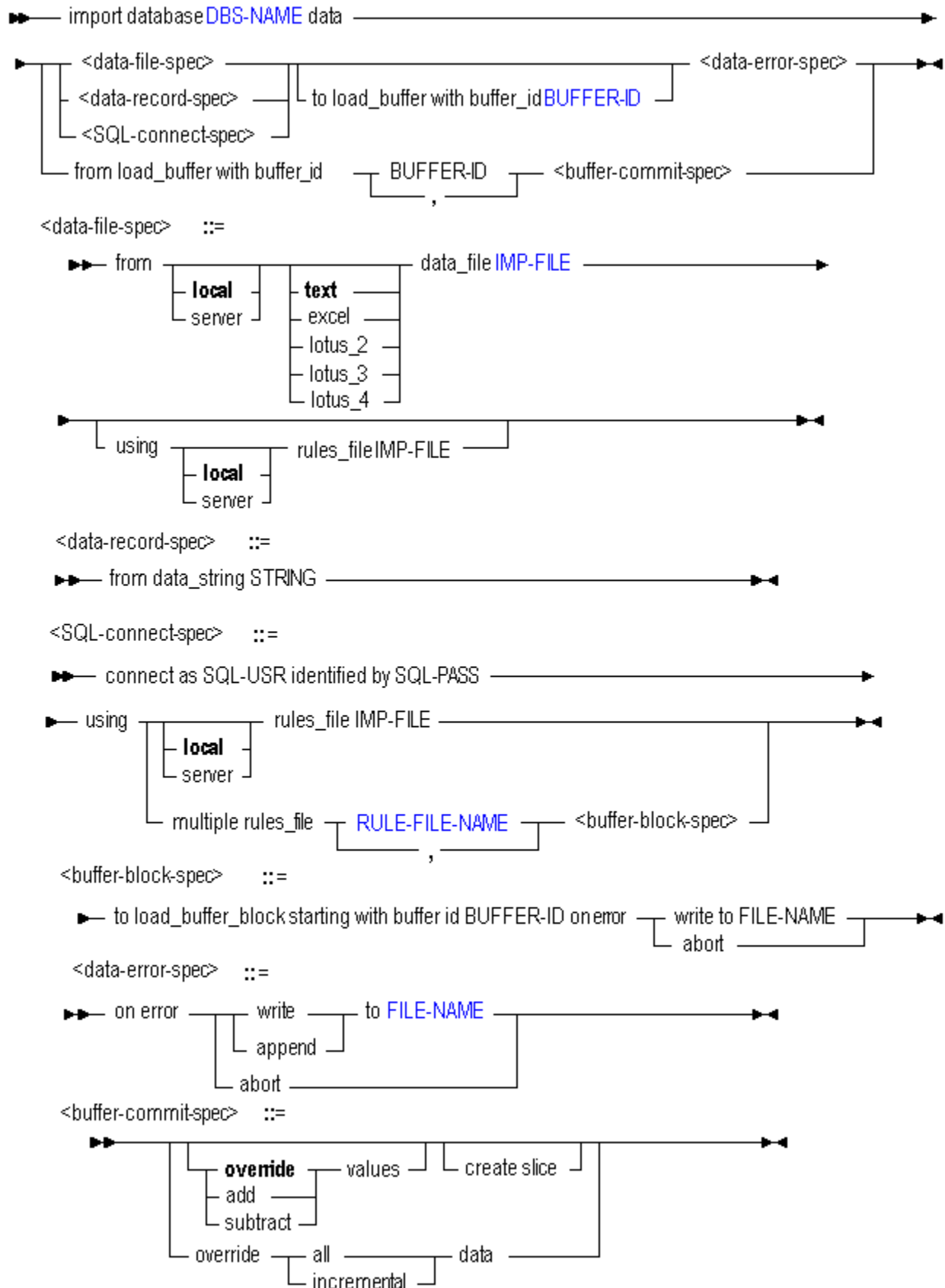
```
export database ASOSamp.Sample using report_file 'my.rep' to data_file 'my.rpt';
```

Import Data (Aggregate Storage)

[Click here for non-aggregate storage version](#)

Import data from text or spreadsheet data files, with or without a rules file. Minimum permission required: Write.

Syntax



Use **import data** in the following ways to load data into an aggregate storage database:

Keyword	Description
<code>import database <db-name> data from...</code>	Specify whether the data import is from a local or server file, and what type of file to import data from.
<code>...using ... rules_file</code>	Import data into the database using a specified rules file.
<code>...<data error spec> (on error...)</code>	Required. Tell Essbase what to do in case of errors during the data load: abort the operation, or write or append to a specified error log.
<code>...<data record spec> from data_string</code>	Load a single data record into the selected database.
<code>...<SQL connect spec> (connect as...)</code>	<p>If you are importing data from an SQL source, provide your SQL user name and password. You must always use a rules file when you load SQL data sources.</p> <p>When loading SQL data into aggregate storage databases, you can use up to eight rules files to load data in parallel by using the multiple rules_file grammar with the grammar specified in <code><buffer-block-spec></code>. Essbase initializes multiple temporary aggregate storage data load buffers (one for each rules file) and, when the data is fully loaded into the buffers, commits the contents of all buffers into the database in one operation.</p> <p>Each rules file must use the same authentication information (SQL user name and password).</p> <p>In the following example, SQL data is loaded from two rules files (<code>rule1.rul</code> and <code>rule2.rul</code>):</p> <pre>import database AsoSamp.Sample data connect as TBC identified by 'password' using multiple rules_file 'rule1','rule2' to load_buffer_block starting with buffer_id 100 on error write to "error.txt";</pre> <p>In specifying the list of rules files, use a comma-separated string of rules file names (excluding the <code>.rul</code> extension). The filename for rules files must not exceed eight bytes and the rules files must reside on Essbase Server.</p> <p>In initializing a data load buffer for each rules file, Essbase uses the starting data load buffer ID you specify for the first rules file in the list (for example, ID 100 for <code>rule1</code>) and increments the ID number by one for each subsequent data load buffer (for example, ID 101 for <code>rule2</code>).</p> <p>The ODBC driver you are using must be configured for parallel SQL connections. See the <i>Oracle Essbase SQL Interface Guide</i>.</p> <p>Note: Performing multiple SQL data loads in parallel to aggregate storage databases is different than using the <code>to load_buffer with buffer_id</code> grammar to load data into a buffer, and then using the <code>from load_buffer with buffer_id</code> grammar to explicitly commit the buffer contents to the database. For more information on aggregate storage data load buffers, see the <i>Oracle Essbase Database Administrator's Guide</i>.</p>
<code>...to load_buffer with buffer_id</code>	If you are importing data from multiple data files to an aggregate storage database, you can import to a buffer first, in order to make the data import operation more efficient.
<code>...from load_buffer with buffer_id</code>	If you are importing data from multiple data files to an aggregate storage database, you can import from a data load buffer in order to make the data import operation more efficient.

Keyword	Description
...from load_buffer with buffer_id...values	Specify whether you want to add to existing values, subtract from existing values, or override existing values when committing the contents of the specified data load buffer to the database.
...from load_buffer with buffer_id...create slice	Commit the contents of the specified data load buffer to the database by creating a new data slice.
...from load_buffer with buffer_id override all data	Remove the current contents of the database and replace the database with the contents of the specified data load buffer.
...from load_buffer with buffer_id override incremental data	Remove the current contents of all incremental data slices in the database and create a new data slice with the contents of the specified data load buffer. The new data is created with the data load property "add values" (aggregate_sum). If there are duplicate cells between the new data and the primary slice, their values are added together when you query for them.

Notes

- This statement requires that the database is started.
- When using the import statement, you must specify what should happen in case of an error.
- To import from a SQL data source, you must connect as the relational user name and use a rules file.

Example

```
import database asosamp.sample data from data_file '$ARBORPATH\app\asosamp\sample\
\dataload.txt' using rules_file '$ARBORPATH\app\asosamp\sample\dataload.rul' on
error abort;
```

Loads data into the ASOSamp.Sample database.

```
import database AsoSamp.Sample data from load_buffer with buffer_id 1;
```

Commits the contents of a specified data load buffer to the AsoSamp.Sample database.

```
import database AsoSamp.Sample data from load_buffer with buffer_id 1, 2;
```

Commits the contents of multiple data load buffers (buffer_id 1 and buffer_id 2) to the AsoSamp.Sample database.

```
import database AsoSamp.Sample data from load_buffer with buffer_id 1 add values;
```

Commits the contents of a specified data load buffer to the AsoSamp.Sample database by adding values.

```
import database AsoSamp.Sample data from load_buffer with buffer_id 1 override values
create slice;
```

Commits the contents of the specified data load buffer into a new data slice in the AsoSamp.Sample database.

```
import database AsoSamp.Sample data from load_buffer with buffer_id 1 override all data;
```

Replaces the contents of the AsoSamp.Sample database with the contents of the specified data load buffer.

```
import database AsoSamp.Sample data from load_buffer with buffer_id 1 override incremental data;
```

Replaces the contents of all incremental data slices in the AsoSamp.Sample database by creating a new data slice with the contents of the specified data load buffer. The new data is created with the data load property "add values" (aggregate_sum). If there are duplicate cells between the new data and the primary slice, their values are added together when you query for them.

See [“Loading Data Using Buffers” on page 921](#).

Query Application (Aggregate Storage)

Get information about the current state of the application.

This statement requires the application to be started. This statement is only applicable for applications using aggregate storage mode.

Syntax

```
query application APP-NAME {
  get cache_size
  list aggregate_storage storage_info
}
```

Example

The following MaxL statement:

```
query application sample get cache_size;
```

returns the maximum size (in kilobytes) to which the aggregate storage cache may grow.

The following MaxL statement:

```
query application asoapp list aggregate_storage storage_info;
```

returns the following information:

Output Columns	Description
Cache hit ratio	Ratio of the number of requests answered from aggregate storage cache as opposed to from the hard disk.
Current cache size (KB)	The current size of the aggregate storage cache. See description for current cache size limit (KB).
Current cache size limit (KB)	The maximum size (in kilobytes) to which the aggregate storage cache may grow.
Page reads since last startup	Number of data blocks (pages) read from disk since the last time the application was started.
Page writes since last startup	Number of data blocks (pages) written to disk since the last time the application was started.
Page size (KB)	Size of the data block (page) in kilobytes.

Output Columns	Description
Disk space allocated for data (KB)	Total space used by all disk files in the default tablespace.
Disk space used by data (KB)	Total space actually in use within the disk files in the default tablespace (some space within files may be free).
Temporary disk space allocated (KB)	Total space used by all disk files in the temp tablespace.
Temporary disk space used (KB)	Total space actually in use within the disk files in the temp tablespace (some space within files may be free).

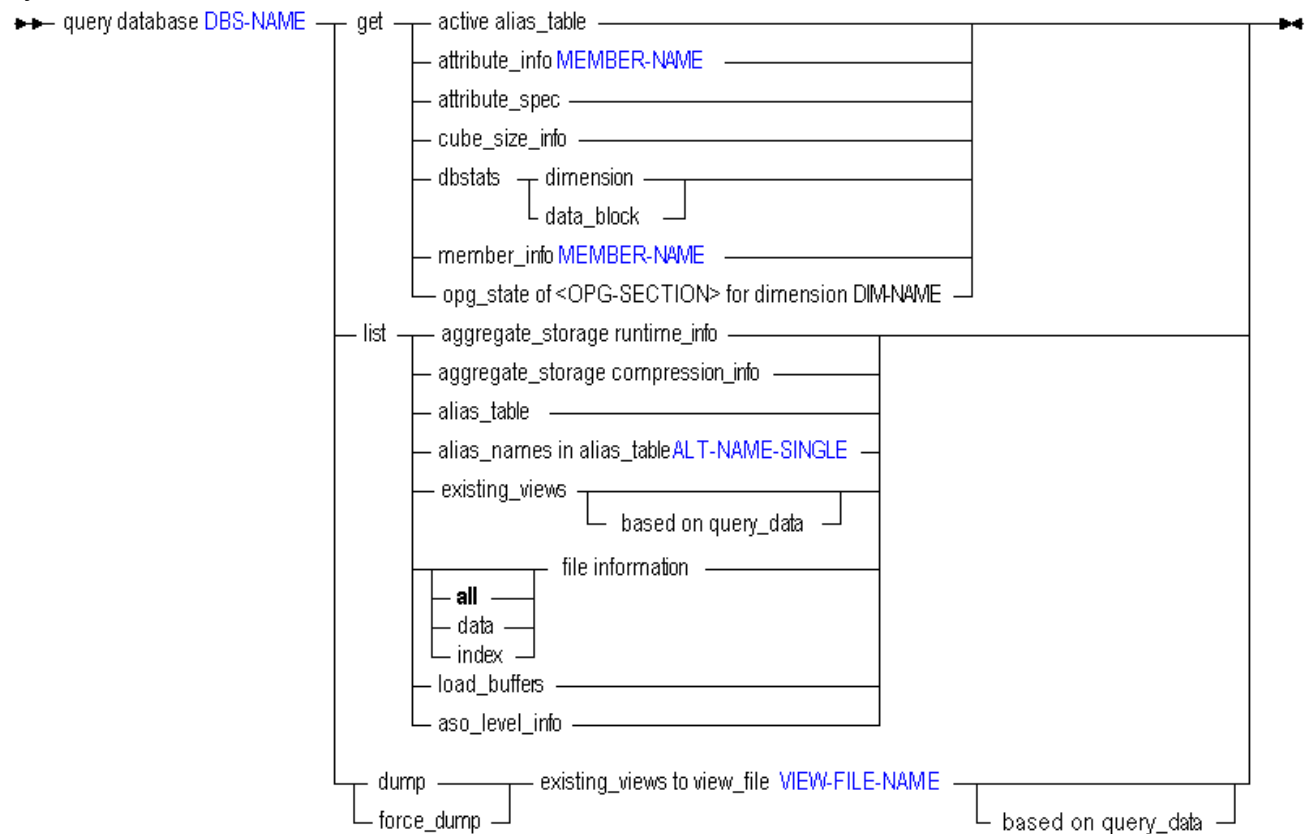
Query Database (Aggregate Storage)

[Click here for non-aggregate storage version](#)

Get advanced information about the current state of the database.

Minimum permission required: Read. This statement requires the database to be started.

Syntax



You can query for database information in the following ways using **query database**:

Keyword	Description
get active alias_table	Display the active alias table for the user issuing the statement.

Keyword**Description**

get attribute_info Get attribute member, dimension, and name information for the specified attribute member.

get attribute_spec Display the current attribute specifications for the database. These specifications include attribute member name format, Attribute Calculation dimension member names, Boolean and date member names, and numeric range specifications. These settings are defined in Outline Editor.

get cube_size_info Display information about input data size, aggregated data size, and number of queries tracked (when query tracking is enabled).

This statement returns the output listed in the following table:

Column Name	Contents
input_data_size_cells	Number of input-level cells in the cube.
input_data_size_bytes	Number of bytes used by the input-level data (approximate).
aggregate_data_size_cells	Total number of cells in all aggregate views in the cube.
aggregate_data_size_bytes	Number of bytes used by the aggregate cells (approximate).
kernel_queries_tracked	Number of kernel queries executed since the last time query tracking was enabled or query tracking information was reset.
total_query_cost	Total cost of all queries executed since the last time query tracking information was reset.
query_tracking_enabled	<p>Values: True or False. Tells whether user retrieval statistics are being collected for the aggregate storage database. The statistics can be used by the following MaxL statements for query-based view optimization:</p> <ul style="list-style-type: none"> ● query database <db-name> list existing_views ● execute aggregate process ● execute aggregate selection <p>Query tracking is disabled by default.</p>

get dbstats dimension Get information about dimensions.

The `index_type` field values are numeric, and translate as follows:

0	Dense
1	Sparse
3	None (database is aggregate storage)

get dbstats data_block Get information about data blocks. The information returned has little relevance to aggregate storage databases.

Keyword**Description**

get member_info <MEMBER-NAME> Get information on a specific member.

Output

The **unary_type** field values are numeric, and translate as follows:

0	Add
1	Subtract
2	Multiply
3	Divide
4	Percent
5	NoRollUp

The **member_tag_type** field values translate as follows:

0	SkipNone
16384	SkipMissing
32768	SkipZero
49152	SkipBoth
1	BalFirst
2	BalLast
4	TwoPass
8	Average
64	Expense

Variations are possible. The field value consists of one of the first four "skip" values plus any/all/none of the last five values. Some examples:

0	SkipNone
77	SkipNone, BalFirst, TwoPass, Average, Expense
16385	SkipMissing and BalFirst

The first four "skip" values are base values, and added to them are combinations of 1, 2, 4, 8, and 64.

The **status** field values are hexadecimal, and translate as follows:

0	Normal
1	Never Share
2	Label
4	Refer Share
8	Refer Share (with different name)
16	Implicit share
32	Virtual Member (stored)
64	Virtual Member (not stored)
2048	Attribute
32768	Referred

get opg_state of member_data

Display outline navigational information (for example, parent, child, or sibling), fixed-length information (for example, the line aggregation symbol or the number of children), and text strings (for example, member names or aliases).

See [“Outline Paging Dimension Statistics” on page 904](#) for a description of the output.

Keyword	Description
get opg_state of member_name_namespace	<p>Display information that matches member names to internal member identifiers (one section per database, thus the information for all dimensions is the same).</p> <p>See “Outline Paging Dimension Statistics” on page 904 for a description of the output.</p>
get opg_state of member_formula	<p>Display all formulas for the dimension.</p> <p>See “Outline Paging Dimension Statistics” on page 904 for a description of the output.</p>
get opg_state of member_UDA	<p>Display all user defined attributes (UDAs) for the dimension.</p> <p>See “Outline Paging Dimension Statistics” on page 904 for a description of the output.</p>
get opg_state of member_UDA_namespace	<p>Display information that matches UDAs to internal member identifiers.</p> <p>See “Outline Paging Dimension Statistics” on page 904 for a description of the output.</p>
get opg_state of attribute_to_base_member_association	<p>Display information that identifies the attribute member associated with each base member of the dimension.</p> <p>See “Outline Paging Dimension Statistics” on page 904 for a description of the output.</p>
get opg_state of member_comment	<p>Display all member comments for the dimension.</p> <p>See “Outline Paging Dimension Statistics” on page 904 for a description of the output.</p>
get opg_state of member_alias_namespace	<p>Display information that matches member alias names to internal member identifiers (one section per alias table, thus the information for all dimensions is the same).</p> <p>See “Outline Paging Dimension Statistics” on page 904 for a description of the output.</p>
list aggregate_storage runtime_info	<p>Display runtime statistics about the aggregate storage database. For a description of the output returned by this statement, see “Aggregate Storage Runtime Statistics” on page 905.</p>

Keyword**Description**

list aggregate_storage compression_info

Display estimated compression for aggregate storage databases when different dimensions are hypothetically used as the accounts dimension. These estimates can help you choose the best dimension to use as the accounts dimension.

In aggregate storage databases, the accounts dimension enables database compression. A good candidate for an accounts dimension is one that optimizes data compression and maintains retrieval performance.

This statement returns the following output:

Column Name	Contents
Accounts Dim	Each dimension name in the database, hypothetically considered to be the accounts dimension.
Stored Level 0 Members	The number of leaf-level members in the dimension. A large number of stored level-0 members in a dimension indicates that it may not perform well as an accounts dimension.
Average Bundle Fill	Estimated average number of values per accounts dimension bundle. Choosing an accounts dimension that has a higher average bundle fill means that the database compresses better.
Average Value Length	Estimated average number of bytes required to store a value. Dimensions with a smaller average value length compress the database better.
Expected Level 0 Size (MB)	<p>Estimated size of of the compressed database, in megabytes. A smaller expected level-0 size indicates that choosing this dimension enables better compression.</p> <p>Except for the scenario in which there is no accounts dimension (<None>), all estimates assume that all pages are compressed. Since compressed pages require additional overhead that uncompressed pages do not, the estimated level-0 database size for some dimensions may be larger than the value for <None>.</p>

list alias_table

Get a list of alias tables that are defined for the database.

list alias_names in alias_table

List the alias names defined in an alias table. Alias tables contain sets of aliases for member names and are stored in the database outline. Use this grammar to see a list of alias names defined in the specified table.

Keyword	Description
list existing_views	<p>Display information about all aggregate views. An aggregate view is a collection of aggregate cells based on the levels of the members within each dimension.</p> <p>The optional based on query_data clause causes the returned query cost information to be based on the collected cost of actual user queries. If this clause is not used, the default assumption is that all possible queries happen with the same probability.</p> <p>To use the based on query_data clause, query tracking must first be enabled. To enable query tracking, use alter database <db-name> enable query tracking.</p> <p>For more information about aggregate views, see the <i>Oracle Essbase Database Administrator's Guide</i>.</p>
list ... file information	<p>Get accurate index and data file information. Provides index and data file names, counts, sizes, and totals, and indicates whether or not each file is presently opened by Essbase. The file size information is accurate. Note that the file size information provided by the Windows operating system for index and data files that reside on NTFS volumes may not be accurate.</p>
list load_buffers	<p>Display a list and description of the data load buffers that exist on an aggregate storage database. See “Using Aggregate Storage Data Load Buffers” on page 923.</p>
list aso_level_info	<p>Display the aggregation level count for each real dimension in the outline. Aggregation level count is the total number of aggregation levels in a real dimension (including associated attribute dimensions) that exist on an aggregate storage database.</p>
dump force_dump existing views...	<p>Saves existing views of this database to an aggregation script. This action requires a minimum permission of execute (“Execute” on page 635).</p> <p>If the specified script name already exists, you can use the force_dump keyword to overwrite it; otherwise, an error is returned if the file name already exists.</p> <p>If the based on query_data phrase is used, the view selection that is saved will be based on previously collected query-tracking data. You must have enabled query tracking to use this option. For more information about query tracking, see the based on query_data description in execute aggregate selection. See also the <i>Oracle Essbase Database Administrator's Guide</i>.</p>

Example

```
query database Asosamp.Sample list load_buffers;
```

Display a list and description of the data load buffers that exist on Asosamp.Sample.

Outline Paging Dimension Statistics

The following columns are the output of the MaxL statement beginning with [query database DBS-NAME get opg_state](#).

This statement is only applicable to databases using aggregate storage.

Column Name	Contents
version	The version of the outline paging section (a Berkeley DB database).
unique_keys	The number of unique keys in the outline paging section.
key/data_pairs	The number of key/data pairs in the outline paging section.
page_size	The page size (in bytes) of the underlying database.
minimum_keys_per_page	The minimum number of keys per page.
length of fixed_length_records	The length of the fixed-length records (only available when the outline paging section is a Recno database).
padding_byte_value_for_fixed_length_columns	The padding byte value for fixed-length records.
levels	Number of levels in the underlying database corresponding to the outline paging section.
internal_pages	Number of internal pages in the underlying database.
leaf_pages	Number of leaf pages in the underlying database.
duplicate_pages	Number of duplicate pages in the underlying database.
overflow_pages	Number of overflow pages in the underlying database.
pages_on_free_list	Number of pages on the free list in the underlying database.
bytes_free_in_internal_pages	Number of bytes free in internal pages of the underlying database.
bytes_free_in_leaf_pages	Number of bytes free in leaf pages of the underlying database.
bytes_free_in_duplicate_pages	Number of bytes free in duplicate pages of the underlying database.
bytes_free_in_overflow_pages	Number of bytes free in overflow pages of the underlying database.

Aggregate Storage Runtime Statistics

Statistics per Dimension

The following MaxL statement:

```
query database asoapp.asodb list aggregate_storage runtime_info;
```

Returns output which includes the following lines:

```
parameter                                value
+-----+-----+
Dimension [Year] has [3] levels, bits used      4
Dimension [Measures] has [1] levels, bits        4
Dimension [Product] has [3] levels, bits u       5
Dimension [Market] has [3] levels, bits us       5
Dimension [Scenario] has [1] levels, bits        2
...
```

For each dimension, the following statistics are shown:

- The name of the dimension.
- How many stored levels the dimension has, in the aggregate storage perspective. Not all levels are stored in aggregate storage databases; some are virtual levels.
- The number of bits being used in the key for the dimension.

Each cell in an aggregate storage database is stored as a key/value pair. The key length is 8 bytes or a multiple of 8 bytes; for example, 8, 16, 24.

Each key corresponds to a numeric value in the database. The statistics shown above report key lengths in bytes and the number of bits used per key. How many bits each dimension uses in the dimensional key is shown in the value column for each dimension.

How many bits used in each key may amount to less than the bytes needed for physical storage of the key. As an example where this knowledge might be useful, consider a case in which a key is using 65 bits. If you can reduce the key length by one bit to 64, then you can have then key length be 8 bytes instead of 9, an improvement which reduces the overall size of the databse. Another use for these statistics might be to examine them to see how much you gain from removing any particular dimension.

Statistics for the Whole Database

The same MaxL statement used above also returns the following lines in its output:

```
parameter                                value
+-----+-----+
...
Max. key length (bits)                    20
Max. key length (bytes)                   0
Number of input-level cells               0
Number of aggregates                      0
Number of aggregate cells                 0
Size of the input level data (KB)
Size of the aggregate data (KB)
```

The whole-database statistics are described in the following table.

Column Name	Description
Max. key length (bits)	The sum of all the bits used by each dimension. For example, there are 20 bits in the key used for dimensions, and the first 4 are used by Year.
Max. key length (bytes)	How many bytes the key uses per cell.
Number of input-level cells	The number of existing level-0 cells in the database.
Number of aggregates	The number of aggregate views in the database.
Number of aggregate cells	The number of cells stored in the database's aggregate views.
Size of the input level data (KB)	The total disk space used by input-level data.
Size of the aggregate data (KB)	The total disk space occupied by aggregate cells.

For input-level and aggregate cells, the above statistics show

1. Number of cells
2. Disk space occupied by those cells

Because Essbase uses compression, these statistics are useful because it is not always possible to derive disk size based on the number of cells.

MaxL Statements for Data Mining

The following are the MaxL statements for data mining.

Data Mining Algorithms

The following are the statements to create, delete, and display information about data-mining algorithms.

[create algorithm](#)

[drop algorithm](#)

[display algorithm](#)

Data Mining Transformations

The following are the statements to create, delete, and display information about data-mining transformations.

[create transformation](#)

[drop transformation](#)

[display transformation](#)

Mining Models

The following are the statements to delete and display information about data-mining models.

[create model](#)

[display model](#)

[drop model](#)

[export model](#)

Mining Results

The following are the statements to create, delete, and display information about data-mining results.

[create mining result](#)

[display mining result](#)

[drop mining result](#)

Mining Task Templates

The following are the statements to create, delete, and display information about data-mining templates.

[create mining task template](#)

[drop mining task template](#)

[display mining task template](#)

Mining Sessions

The following statements pertain to current data-mining sessions.

[display mining session](#)

[alter system stop mining session](#)

[query database \(<db-name> score miner with mining session ...\)](#)

Data Mining Statements Listed by Verbs

Create

[Create Algorithm](#)

[Create Mining Result](#)

[Create Mining Task Template](#)

[Create Model](#)

[Create Transformation](#)

Display

[Display Algorithm](#)

[Display Mining Result](#)

[Display Mining Session](#)

[Display Mining Task Template](#)

Display Mining Model
Display Transformation

Drop

Drop Algorithm
Drop Mining Result
Drop Mining Task Template
Drop Model
Drop Transformation

Export

Export Mining Model

Data Mining Statements Listed by Objects

Algorithm

Create Algorithm
Display Algorithm
Drop Algorithm

Model

Create Model
Display Mining Model
Drop Model
Export Mining Model

Result

Create Mining Result
Display Mining Result
Drop Mining Result

Session

Display Mining Session

Task

Create Mining Task Template
Display Mining Task Template

Data Mining Statements

The following topics describe the Data Mining statements.

Create Algorithm

Register a new data mining algorithm for use with Essbase.

The Essbase Data Mining Framework provides a set of built-in algorithms. This statement enables you to register additional algorithms from third-party vendors.

Before you register an algorithm, be certain that:

- A Java wrapper has been created for the algorithm.
- The Java code for the algorithm has been compiled and a JAR file containing the class has been created.
- The JAR file containing the class is contained in the *ESSBASEPATH\java\udf* directory.

Minimum permission required: Administrator.

Syntax

```
▶▶ create _____ algorithm ALG-NAME as ALG-CLASS ▶▶  
    |  
    | or replace |
```

You can register an algorithm in the following ways:

Keyword	Description
create algorithm	Register an algorithm. Algorithm names are case sensitive.
create or replace algorithm	Register an algorithm, or replace an existing algorithm of the same name. Algorithm names are case sensitive.

Example

```
create algorithm Regression as 'com.hyperion.essbase.algorithms.Registration';
```

Registers an algorithm named Regression for use with the Data Mining Framework. The class, `com.hyperion.essbase.algorithms.Registration`, implements this algorithm.

```
create or replace algorithm Clustering as 'com.hyperion.essbase.algorithms.clustering';
```

Registers an algorithm named Clustering for use with the Data Mining Framework. The class, `com.hyperion.essbase.algorithms.Clustering`, implements this algorithm. If an algorithm named Clustering already exists, the new version replaces it.

Create Mining Task Template

Create a data mining task template. Any mining task specification can be saved as a template. This template can be later retrieved, modified and invoked as a task.

Minimum permission required: No minimum permission needed to create a template. Administrator or template owner required to replace a template.

Syntax

```
create [ or replace ] mining task template TASK-NAME [ as TASK-XML-STRING ] [ from [ local | server ] file FILE-NAME ]
```

Example

```
create or replace mining task template 'AssocRule' from server file 't1xAssocRule.dmb';
```

Create Model

Create a data-mining model using either a data-mining template or a model from another system exported to PMML.

Permission required: Database Manager. To replace an existing model, you must be an Administrator or the owner of the model.

Syntax

```
create [ or replace ] model MODEL-NAME on database DBS-NAME [ using algorithm ALG-NAME ] [ from [ local | server ] pmml_file FILE-NAME ] [ as TASK-XML-STRING ]
```

Notes

PMML stands for Predictive Model Markup Language, a standard exchange format developed by the Data Mining Group (<http://www.dmg.org>).

Example

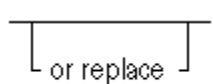
```
create model 'm2xModel' on database Sample.Basic using algorithm 'Regression' from local pmml_file 'e:/essbase/java/dmf/pmml/m2xRegression.pmml' ;
```

Create Mining Result

Initiate a data-mining session, using previously created models.

Permission required: execute calculation privilege. To replace an existing result, you must be an Administrator or the owner of the result.

Syntax

►► create  mining result **RESULT-NAME** with mode **RESULT-MODE** →
 └─ or replace ─┘
 └─ with mode ─┘
◀ on database **DBS-NAME** using model **MODEL-NAME** as **TASK-XML-STRING** →►

Example

The following example creates a mining result from an XML string which defines model data. The XML string should not be written manually; it must be obtained from a display statement. For more information see the definition of “**TASK-XML-STRING**” on page 806.

The easiest way to change the mining task specification is to use the Administration Services Data Mining Wizard. See "About Data Mining" in the *Oracle Essbase Administration Services Online Help* for information.

If you do make changes to the template you obtain from the server, restrict changes to the attributes of the <task> element, the <information> element, and the contents of the <expression> elements.

```
create or replace mining result 'rlxRegression'
with mode 'apply' on database 'DMDemo'. 'Basic' using model 'mlxRegression' as '<task
algorithm=\'MultivariateRegression\' class=\'com.hyperion.essbase.algorithms.Regression
\' mode=\'apply\' model=\'mlxRegression\' output=\'rlxRegression\' owner=\'\' pmml=\'\'
source=\'DMDemo.Basic\'>
<information>
Execution of this task produces forecasted target values for new predictors values. The
algorithm uses the previously constructed regression coefficients to output target
results for each input predictor vector. The forecast is based on the formula: target =
(slope * predictor) + intercept, where the (*) operation stands for dot product.
</information>
<modelInfo>
This is a test model
</modelInfo>
<setting name=\'missingTreatment\' value=\'NaN\'>
<information>
Define missing treatment by the framework
</information>
</setting>
<setting name=\'language\' value=\'mdx\'>
<information>
Choose language for expressions
</information>
</setting>
<accessor mode=\'read\' name=\'Predictor\' type=\'numerical\'>
<information>
References mining attributes used as predictors
</information>
<domain name=\'Predictor\' type=\'attribute\'>
<information>
Defines the locations of the predictor attributes. Traverses the coordinates of the
predictor vector.
</information>
<expression>
{[Television], [DVD], [VCR]}
</expression>
```



```

</domain>
<domain name='Sequence\' type='sequence\'>
<information>
Traverses the sequence of predictor/target values
</information>
<expression>
{[Jan 1].Level.Members}
</expression>
</domain>
<domain name='External\' type='external\'>
<information>
Determines the scope of the model
</information>
<expression>
{[East].Children}
</expression>
</domain>
<anchor>
<information>
Determines additional restrictions
</information>
<expression>
{([2001], [Actual], [Sales])}
</expression>
</anchor>
</accessor>
<accessor mode='write\' name='Target\' type='numerical\'>
<information>
This mining attribute is used as the target
</information>
<domain name='Target\' size='1\' type='attribute\'>
<information>
Defines the location of the single target value
</information>
<expression>
{[Camera]}
</expression>
</domain>
<domain name='Sequence\' type='sequence\'>
<information>
Traverses the sequence of predictor/target values
</information>
<expression>
{[Jan 1].Level.Members}
</expression>
</domain>
<domain name='External\' type='external\'>
<information>
Determines the scope of the model
</information>
<expression>
{[East].Children}
</expression>
</domain>
<anchor>
<information>
Determines additional restrictions

```

```

</information>
<expression>
{([2001], [Actual], [Sales])}
</expression>
</anchor>
</accessor>
</task>';

```

Create Transformation

Register a data mining transformation.

Transformations can be applied to any accessor within a data mining task. If the accessor is reading from the Essbase database, the transformation is applied to the value of each cell before it reaches the data mining algorithm. If the accessor is writing into an Essbase database, the transformation is applied to the data before it is written into its cell.

For example, a transformation could be registered that squares the result of a mining operation, or offsets the result by a set amount.

The Essbase Data Mining Framework provides a set of built-in transformations. This statement enables you to register additional transformations that you develop or obtain from third-party vendors.

Before you register a transformation, be certain that:

- A Java wrapper has been created for the transformation.
- The Java code for the transformation has been compiled and a JAR file containing the class has been created.
- The JAR file containing the class is contained in the *ESSBASEPATH*\java\udf directory.

Minimum permission required: Administrator.

Syntax

```

▶▶ create ——— transformation TRANSF-NAME as TRANSF-CLASS —▶▶
      |
      | or replace
      |

```

You can register a transformation in the following ways:

Keyword	Description
create transformation	Register a transformation. Transformation names are case sensitive.
create or replace transformation	Register a transformation, or replace an existing transformation of the same name. Transformation names are case sensitive.

Example

```
create transformation 'Scale' as 'com.hyperion.essbase.transformations.Scale';
```

Registers a transformation named Scale for use with the Data Mining Framework. The class, *com.hyperion.essbase.transformations.Scale*, implements this transformation.

```
create or replace transformation 'Log' as 'com.hyperion.essbase.transformations.Log';
```

Registers a transformation named Log for use with the Data Mining Framework. The class, `com.hyperion.essbase.transformations.log`, implements this transformation. If a transformation named Log already exists, the new version replaces it.

Display Algorithm

View a list of registered data mining algorithms.

Minimum permission required: Read.

Syntax

```
▶▶ display algorithm all  
template ALG-NAME with mode ALG-MODE  
ALG-NAME mining task list ▶▶
```

Example

```
display algorithm all;
```

Lists all registered data mining algorithms.

```
display algorithm template Regression as build;
```

Displays the build.

Display Mining Result

View a data-mining result.

Minimum permission required: Read.

Syntax

```
▶▶ display mining result with mode RESULT-MODE  
all  
on database DBS-NAME with mode RESULT-MODE  
RESULT-NAME with mode RESULT-MODE attribute  
RESULT-NAME data with mode RESULT-MODE with attribute all  
RESULT-ACCESSOR ▶▶
```

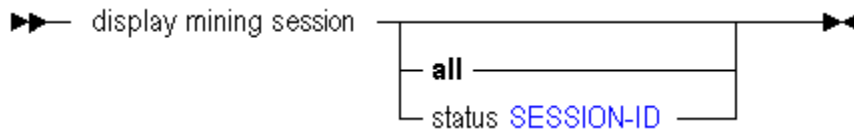
Example

```
display mining result with mode 'apply';
```

Display Mining Session

View a list of current data-mining sessions.

Syntax



Example

```
display mining session;
```

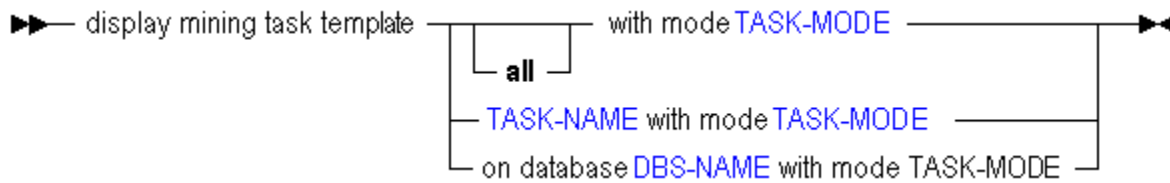
Displays all data-mining sessions.

Display Mining Task Template

View a data-mining task template.

Minimum permission required: Read.

Syntax



Example

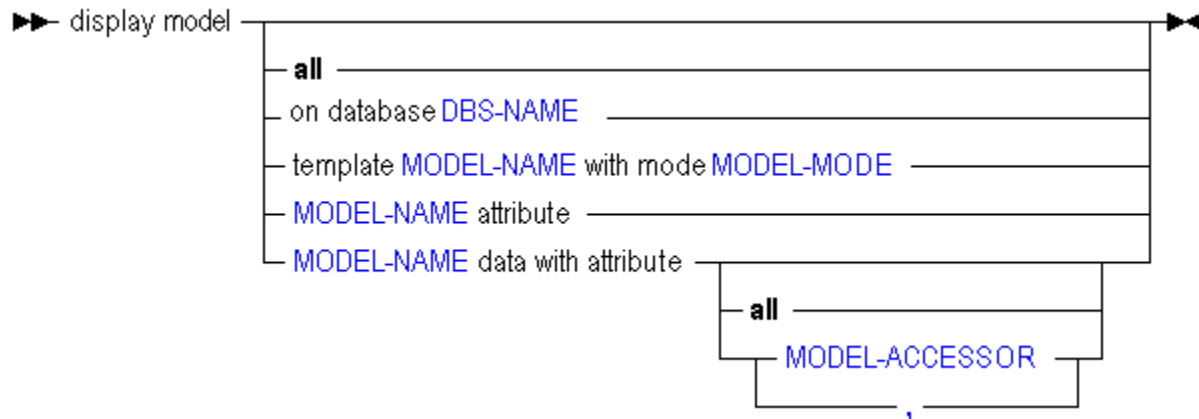
```
display mining task template with mode 'build';
```

Display Mining Model

View a list of data-mining models.

Minimum permission required: Read.

Syntax



You can list information about a model in the following ways:

Keyword	Description
all	List information about all data mining models.
on database	List information about all models that belong to the specified database.
template...	List information about models with the specified name of the specified type (see “ MODEL-MODE ” on page 796).
attribute	Show all accessors for the specified model.
data with attribute all	Show all accessors for the specified model.
data with attribute	Show the specified accessor for the specified model.

Example

```
display model all;
```

Lists information about all data mining models.

```
display model on database Sample.Basic;
```

Displays only those models that belong to the Sample.Basic database.

```
display model template m1AssocRule.dmm with mode apply;
```

Lists information about the m1AssocRule apply model.

```
display model m1Regression.dmm attribute;
```

Shows all accessors for the m1Regression model.

```
display model m1Cluster.dmm data with attribute predictor;
```

Shows the predictor accessor for the m1Cluster model.

Display Transformation

View a list of data-mining transformations.

Minimum permission required: Read.

Syntax

```
▶▶— display transformation —▶▶
```

all
template TRANSF-NAME

You can list information about transformations in the following ways:

Keyword Description

all	List information about all data mining transformations.
template	Obtain the template to use for the transformation. This information must be inserted into the desired accessor's XML element within the algorithm/model template.

Example

```
display transformation;
```

Lists information about all the registered data mining transformations.

```
display transformation template 'Scale';
```

Lists information about the transformation named Scale.

Drop Algorithm

Delete a data-mining-algorithm registration. Once an algorithm is deleted, it is no longer available for use in Essbase.

Minimum permission required: Administrator.

Built-in algorithms provided with Essbase cannot be deleted.

Syntax

```
▶▶— drop algorithm ALG-NAME —▶▶
```

Example

```
drop algorithm Regression;
```

Deletes the algorithm Regression so it is no longer available for use in Essbase.

Drop Mining Result

Delete a data-mining result. Once it is deleted, it is no longer available for use in Essbase.

Minimum permission required: Administrator (or model owner).

Syntax

▶▶— drop mining result **RESULT-NAME** with mode **RESULT-MODE** —▶▶

Example

```
drop mining result 'r1xAssocRule' with mode 'apply';
```

Drop Mining Task Template

Delete a data-mining task template. Once a task template is deleted, it is no longer available for use in Essbase.

Minimum permission required: Administrator or template owner.

Syntax

▶▶— drop mining task template **TASK-NAME** with mode **TASK-MODE** —▶▶

Example

```
ddrop mining task template 'QAAssocRule' with mode 'build';
```

Drop Model

Delete a data-mining model. Once the model is deleted, it is no longer available for use in Essbase.

Minimum permission required: Administrator (or model owner).

Syntax

▶▶— drop model **MODEL-NAME** —▶▶

Example

```
drop model m1Regression.dmm;
```

Deletes the m1Regression model so it is no longer available for use in Essbase.

Drop Transformation

Delete a data-mining transformation. Once a transformation is deleted, it is no longer available for use in Essbase.

Minimum permission required: Administrator.

Built-in transformations provided with Essbase cannot be deleted.

Syntax

▶▶— drop transformation **TRANSF-NAME** —▶▶

Example

```
drop transformation 'Scale';
```

Deletes the transformation Scale so it is no longer available for use in Essbase.

Export Mining Model

Export a data-mining model to PMML format.

Minimum permission required: Administrator or model owner.

Syntax

```
▶▶— export model MODEL-NAME to local pmml_file FILE-NAME —▶▶
```

Notes

The export file is an XML file in PMML format which contains the model data. PMML stands for Predictive Model Markup Language, a standard exchange format developed by the Data Mining Group (<http://www.dmg.org>).

Example

```
export model 'm2xModel' to local pmml_file 'e:/essbase/java/dmf/pmml/m2xModel.pmml';
```

MaxL Use Cases

Creating an Aggregate Storage Sample Using MaxL

Related MaxL statements: [create application](#), [create database](#), [create outline](#), [alter database](#), [import data](#), [execute aggregate process](#),

The following sample MaxL script creates an aggregate storage application and database based on Sample Basic.

```
login $1 $2;

spool on to 'maxl_log.txt';

create or replace application Sample2 using aggregate_storage
comment 'aggregate storage version of Sample';

create database Sample2.Basic2
comment 'aggregate storage version of Sample Basic';

create or replace outline on aggregate_storage database Sample2.Basic2
as outline on database sample.basic;

alter database Sample2.Basic2 initialize load buffer with buffer_id 1;

import database Sample2.Basic2 data
from server data_file 'C:\\Hyperion\\products\\Essbase\\EssbaseServer\\app\\Sample2\\
\\Basic2\\calcdat.txt'
to load_buffer with buffer_id 1
on error abort;
```



```
import database Sample2.Basic2 data from load_buffer with buffer_id 1;

execute aggregate process on database Sample2.Basic2
  stopping when total_size exceeds 1.9;

spool off;

logout;
```

Loading Data Using Buffers

Related MaxL Statements

- [Alter Database \(Aggregate Storage\)](#)
- [Query Database \(Aggregate Storage\)](#)
- [Import Data \(Aggregate Storage\)](#)

If you use multiple [Import Data \(Aggregate Storage\)](#) statements to load data values to aggregate storage databases, you can significantly improve performance by loading values to a temporary data load buffer first, with a final write to storage after all data sources have been read.

While the data load buffer exists in memory, you cannot build aggregations or merge slices, as these operations are resource-intensive. You can, however, load data to other data load buffers, and perform queries and other operations on the database. There might be a brief wait for queries, until the full data set is committed to the database and aggregations are created.

The data load buffer exists in memory until the buffer contents are committed to the database or the application is restarted, at which time the buffer is destroyed. Even if the commit operation fails, the buffer is destroyed and the data is not loaded into the database.

Multiple data load buffers can exist on a single aggregate storage database. To save time, you can load data into multiple data load buffers at the same time by using separate MaxL Shell sessions. Although only one data load commit operation on a database can be active at any time, you can commit multiple data load buffers in the same commit operation, which is faster than committing buffers individually.

You can query the database for a list and description of the data load buffers that exist on an aggregate storage database. See [“Using Aggregate Storage Data Load Buffers” on page 923](#).

Examples:

- [Example: Load Multiple Data Sources into a Single Data Load Buffer](#)
- [Example: Perform Multiple Data Loads in Parallel](#)

Example: Load Multiple Data Sources into a Single Data Load Buffer

Assume there are three data files that need to be imported. With aggregate storage databases, data loads are most efficient when all data files are loaded using one import operation. Therefore, load buffers are useful when loading more than one data file.

1. Use [Alter Database \(Aggregate Storage\)](#) to create a load buffer.

```
alter database ASOSamp.Sample
initialize load_buffer with buffer_id 1;
```

2. Load data into the buffer, using the [Import Data \(Aggregate Storage\)](#) statement.

```
import database ASOSamp.Sample data
from server data_file 'file_1'
to load_buffer with buffer_id 1
on error abort;
```

```
import database ASOSamp.Sample data
from server data_file 'file_2'
to load_buffer with buffer_id 1
on error abort;
```

```
import database ASOSamp.Sample data
from server data_file 'file_3'
to load_buffer with buffer_id 1
on error abort;
```

3. Move the data from the buffer into the database.

```
import database ASOSamp.Sample data
from load_buffer with buffer_id 1;
```

The data-load buffer is implicitly destroyed.

4. Assume that in Step 2, after loading 'file_2' into the load buffer, you decided not to load the data. Because the data is in a buffer and not yet in the database, you would simply use [Alter Database \(Aggregate Storage\)](#) to destroy the buffer without moving the data to the database.

```
alter database ASOSamp.Sample
destroy load_buffer with buffer_id 1;
```

Example: Perform Multiple Data Loads in Parallel

1. In one MaxL Shell session, load data into a buffer with an ID of 1:

```
alter database AsoSamp.Sample
initialize load_buffer with buffer_id 1 resource_usage 0.5;
```

```
import database AsoSamp.Sample data
from data_file "dataload1.txt"
to load_buffer with buffer_id 1
on error abort;
```

2. Simultaneously, in another MaxL Shell session, load data into a buffer with an ID of 2:

```
alter database AsoSamp.Sample
initialize load_buffer with buffer_id 2 resource_usage 0.5;
```

```
import database AsoSamp.Sample data
from data_file "dataload2.txt"
to load_buffer with buffer_id 2
on error abort;
```

3. When the data is fully loaded into the data load buffers, use one MaxL statement to commit the contents of both buffers into the database by using a comma separated list of buffer IDs:

```
import database AsoSamp.Sample data
from load_buffer with buffer_id 1, 2;
```

Using Aggregate Storage Data Load Buffers

Related MaxL Statement:

[Query Database \(Aggregate Storage\)](#)

Use the following MaxL statement to get a list and description of the data load buffers that exist on an aggregate storage database.

```
query database appname.dbname list load_buffers;
```

This statement returns the following information about each existing data load buffer:

Field	Description
buffer_id	ID of a data load buffer (a number between 1 and 4294967296).
internal	A Boolean that specifies whether the data load buffer was created internally by Essbase (TRUE) or by a user (FALSE).
active	A Boolean that specifies whether the data load buffer is currently in use by a data load operation.
resource_usage	The percentage (a number between .01 and 1.0 inclusive) of the aggregate storage cache that the data load buffer is allowed to use.
aggregation method	One of the methods used to combine multiple values for the same cell within the buffer: <ul style="list-style-type: none"> ● AGGREGATE_SUM: Add values when the buffer contains multiple values for the same cell. ● AGGREGATE_USE_LAST: Combine duplicate cells by using the value of the cell that was loaded last into the load buffer.
ignore_missings	A Boolean that specifies whether to ignore #MI values in the incoming data stream.
ignore_zeros	A Boolean that specifies whether to ignore zeros in the incoming data stream.

Specifying Port Numbers in Partition Host Names

You can install multiple agents on a single Windows computer. When multiple agents are installed on a single computer, you can connect to an agent by specifying the host name and the agent port number, in the form: *hostName:agentPort*.

When creating partitions across different ports, you must do the following:

1. Specify the current *hostName:agentPort* when you log in to Essbase. For example, `login partitionuser mypassword on 'localhost:3300';`
2. Specify the target *hostName:agentPort* as part of the create, alter, drop, or refresh partition statement. For example,

```
create or replace transparent partition sampeast.east
area '@CHILDREN("Eastern Region"), @CHILDREN(Qtr1)' sourceArea
to samppart.company at 'localhost:2200'
```

```
as partitionuser identified by mypassword
    area '@CHILDREN(East) @CHILDREN(Qtr1)' targetArea;
```

If you log on to Essbase specifying the agent port, then you must specify the agent port for partition operations. If you do not log in specifying the agent port, then do not specify the agent port for partition operations.

The first DBS-NAME specified in a statement is the local database, and the second DBS-NAME is the remote database. Only the remote (second) DBS-NAME in any partition statement can be specified using an agent port. Therefore, when dealing with multiple agent ports, always put the side of the partition that you aren't logged on to second in the statement, so that you can specify which *hostName:agentPort* it is on.

See Also

[“Using Host Name Aliases When Partitioning” on page 924](#)

Using Host Name Aliases When Partitioning

If you want to use network aliases for the data source or data target names, you must make sure that the aliases are propagated to all computers on your system. Otherwise, use the full server name.

To propagate an alias to all the computers on your system, edit the `/etc/hosts` file (on UNIX systems) or the `%WINDIR%/system32/drivers/etc/hosts` file (on Windows systems), adding an entry with the IP address, followed by the host name, followed by the alias.

For example, if you want to use an alias `abcdefg.hijk.123` for a system with host name `hostname.domainname` having IP address `172.234.23.1`, then the host file entry should be:

```
172.234.23.1 hostname.domainname abcdefg.hijk.123
```

In case of multiple aliases, append the aliases following the hostname. For example, if you want to use multiple aliases `abcdefg.hijk.123` and `lmnopqrs.tuvw.456` for a system with host name `hostname.domainname` having IP address `172.234.23.1`, then the host file entries should be:

```
172.234.23.1 hostname.domainname abcdefg.hijk.123 lmnopqrs.tuvw.456
172.234.23.1 hostname.domainname lmnopqrs.tuvw.456 abcdefg.hijk.123
```

Notes

- Do not use `localhost` as an alias to specify source and target server names.
- The user should have root or admin privileges for the system to edit the `hosts` file.

See Also

[“Specifying Port Numbers in Partition Host Names” on page 923](#)

Partitioning and SSL

The following considerations apply when partitioning in secure (SSL) mode:

- The partition source and target must have the same security protocol; for example, both or neither use SSL.
- To enable Essbase to use SSL connectivity, you must set `ENABLESECUREMODE` to TRUE.
- Consider setting `CLIENTPREFERREDMODE` to SECURE.

If `CLIENTPREFERREDMODE` is not set, or is set to FALSE, but `ENABLESECUREMODE` is set to TRUE, you can securely create and refresh partitions in MaxL by adding `:secure` to the HOST-NAME string. For example,

```
login esbuser esbpassword on "localhost:6423:secure";
```

Forcing Deletion of Partitions

The `force` keyword used at the end of the `drop partition` statement specifies that the source half of a partition definition should be dropped regardless of whether the target half is missing or invalid.

For example, in the following session, assume there is a partition definition between `app1.source` and `app2.target`, but the `app2.target` database has been dropped. An ordinary attempt to drop the partition definition fails:

```
MAXL> drop transparent partition app1.source to app2.target;

OK/INFO - 1053012 - Object source is locked by user system.
OK/INFO - 1051034 - Logging in user System.
OK/INFO - 1051035 - Last login on Friday, January 10, 2005 2:28:09 PM.
ERROR - 1051032 - Database target does not exist.
OK/INFO - 1053013 - Object source unlocked by user system.
OK/INFO - 1051037 - Logging out user system, active for 0 minutes.
```

In the second attempt, the `force` keyword allows the invalid source partition to be dropped:

```
MAXL> drop transparent partition app1.source to app2.target force;

OK/INFO - 1053012 - Object source is locked by user system.
OK/INFO - 1051034 - Logging in user System.
OK/INFO - 1051035 - Last login on Friday, January 10, 2005 2:31:50 PM.
ERROR - 1051032 - Database target does not exist.
OK/INFO - 1051037 - Logging out user system, active for 0 minutes.
OK/INFO - 1053013 - Object source unlocked by user system.
OK/INFO - 1241125 - Partition dropped.
```

Note: The `force` keyword only works to drop a partition definition when the source half of the partition definition remains valid. In other words, if the source database is deleted, the partition cannot be dropped from the dangling target.

Metadata Filtering

Related MaxL statements: [create filter](#), [alter filter](#).

Metadata filtering provides an additional layer of security in addition to data filtering. With metadata filtering, an administrator can remove outline members from a user's view, providing access only to those members that are of interest to the user.

When a filter is used to apply MetaRead permission on a member,

1. Data for all ancestors of that member are hidden from the filter user's view.
2. Data *and* metadata (member names) for all siblings of that member are hidden from the filter user's view.

Example

The following report script for Sample Basic:

```
//Meta02.rep  
  
<COLUMN (Year, Product)  
<CHILDREN Cola  
  
<ROW (Market)  
<ICHILDREN West  
!
```

under normal unfiltered conditions returns

	Year 100-10 Measures Scenario
California	3,498
Oregon	159
Washington	679
Utah	275
Nevada	(18)
West	4,593

But with the following filter granted to an otherwise read-access user,

```
create or replace filter sample.basic.meta02  
  meta_read on ('California','Oregon')  
;
```

the report script then returns:

	Year 100-10 Measures Scenario
California	3,498
Oregon	159
West	#Missing

In summary, MetaRead permission on California and Oregon means that:

1. The affected user can see no data for ancestors of California and Oregon members. West data shows only #Missing (or #NoAccess, in a spreadsheet interface).
2. The affected user can see no sibling metadata (or data) for siblings of California and Oregon. In other words, the user sees only the western states for which the filter gives MetaRead permission.

Overlapping Metadata Filter Definitions

You should define a MetaRead filter using multiple rows only when the affected member set in any given row (the metaread members and their ancestors) has no overlap with MetaRead members in other rows. It is recommended that you specify one dimension per row in filters that contain MetaRead on multiple rows. However, as long as there is no overlap between the ancestors and MetaRead members, it is still valid to specify different member sets of one dimension into multiple MetaRead rows.

For example, in Sample Basic, the following filter definition has overlap conflicts:

Access	Member Specification
MetaRead	California
MetaRead	West

In the first row, applying MetaRead to California has the effect of allowing access to California but blocking access to its ancestors. Therefore, the MetaRead access to West is ignored; users who are assigned this filter will have no access to West.

If you wish to assign MetaRead access to West as well as California, then the appropriate method is to combine them into one row:

Access	Member Specification
MetaRead	California,West

Examples of Triggers

Related MaxL statements: [alter trigger](#), [create trigger](#), [display trigger](#), [drop trigger](#).

The triggers feature is licensed separately from Essbase. The following examples are based on the Sample Basic database.

Note: You cannot define a trigger that requires data from Dynamic Calc members, hybrid analysis members, or members from another partition.

Example 1: Tracking Sales for January

Example 1 tracks the Actual, Sales value for the following month, product, and region:

- January (Year dimension member Jan)
- Colas (Product dimension member 100)
- In the Eastern region (Market dimension member East)

When the current member being calculated is Jan, and when the Actual, Sales value of Colas for January exceeds 20, the example logs an entry in the file `Trigger_jan_Sales`.

```

create or replace trigger Sample.Basic.Trigger_Jan_20
Where
  {(Jan,Sales,[100],East,Actual)}
When
  Jan > 20 AND Is(Year.CurrentMember, Jan)
then spool Trigger_Jan_20
end;

```

Example 2: Tracking Sales for Quarter 1

Example 2 tracks the Actual, Sales value for the following months, product, and region:

- January, February, March (The children of Year dimension member Qtr1)
- Colas (Product dimension member 100)
- In the Eastern region (Market dimension member East)

When the current member being calculated is Jan, Feb or Mar, and when the Actual, Sales value of Colas for any of the the months January, February, or March exceeds 20, the example logs an entry in the file Trigger_Jan_Sales_20, Trigger_Feb_Sales_20, or Trigger_Mar_Sales_20.

```

create or replace trigger Sample.Basic.Trigger_Qtr1_Sales
Where
Crossjoin(
  {Qtr1.children},
  {[Measures].[Sales], [Product].[100], [Market].[East], [Scenario].[Actual]})
)
When
  Year.Jan > 20 and is(Year.currentmember, Jan)
then spool Trigger_Jan_Sales_20
When
  Year.Feb > 20 and is(Year.currentmember, Feb)
then spool Trigger_Feb_Sales_20
When
  Year.Mar > 20 and is(Year.currentmember, Mar)
then spool Trigger_Mar_Sales_20
end;

```

Example 3: Tracking Inventory Level

Example 3 tracks the inventory level for the following product, region, and months:

- Colas (product 100)
- In the eastern region (market East)
- For January, February, and March (the children of Qtr1)

If the inventory of Colas in the eastern region falls below 500,000, the example trigger sends an email to recipient@company.com.

```

create or replace trigger Sample.Basic.Inventory_east
where CrossJoin(
  {[Qtr1].children},
  {[East],[100],[Ending Inventory]})
)
when [Ending Inventory] < 500000 then

```



```
mail ([smtp_server.company.com], [sender@company.com],  
      [recipient@company.com],  
      [Subject of E-Mail])  
end;
```

7

MDX

In This Chapter

Overview of MDX	931
MDX Query Format	932
MDX Syntax and Grammar Rules	932
MDX Operators	963
About MDX Properties	964
MDX Comments	970
MDX Query Limits	971
Aggregate Storage and MDX Outline Formulas.....	973
MDX Functions	989
MDX Function Reference	997

Overview of MDX

MDX is a language-based data analysis mechanism to Essbase databases. MDX exhibits all of the following characteristics:

- Provides advanced data extraction capability
- Provides advanced reporting capability
- Includes functions for identifying and manipulating very specific subsets of data
- Is a data-manipulation language, complementing MaxL DDL (the data-definition language for Essbase)
- Utilizes the platform-independent XML for Analysis specification

MDX is a joint specification of the XML for Analysis founding members. For more information about XML for Analysis, please visit <http://www.xmlforanalysis.com>.

MDX is a language for anyone who needs to develop scripts or applications to query and report against data and metadata in Essbase databases. The following prerequisite knowledge is assumed:

- A working knowledge of the operating system your server uses and the ones your clients use.
- An understanding of Essbase concepts and features.
- Familiarity with XML.

In order for Essbase to receive MDX statements, you must pass the statements to Essbase. To pass statements, use either the MaxL Shell (essmsh) or MDX Script Editor in Administration Services. When using the MaxL Shell, terminate all statements with a semicolon. Results are returned in the form of a grid.

MDX Query Format

Every query using the SELECT statement has the following basic format. Items in [brackets] are optional.

```
[<with_section>]
SELECT [<axis_specification>
      [, <axis_specification>...]]
      [FROM [<cube_specification>]]
      [WHERE [< slicer_specification>]]
```

Item	Description
<with_section>	An optional section, beginning with the keyword WITH, in which you can define referenceable sets or members.
SELECT	A literal keyword that must precede axis specifications.
[<axis_specification> [, <axis_specification>...]]	Any number of comma-separated axis specifications. Axes represent an <i>n</i> dimensional cube schema. Each axis is conceptually a framework for retrieving a data set; for example, one axis could be thought of as a column, and the next could be considered a row. See “MDX Axis Specifications” on page 949 for more information.
FROM	A literal keyword that must precede the cube specification.
<cube_specification>	The name of the database from which to select. If left blank, the current database context is assumed.
WHERE	A literal keyword that must precede the slicer specification, if one is used.
<slicer_specification>	A tuple, member, or set representing any further level of filtering you want done on the results. For example, you may want the entire query to apply only to Actual Sales in the Sample Basic database, excluding budgeted sales. The WHERE clause might look like the following: <code>WHERE ([Scenario].[Actual], [Measures].[Sales])</code>

MDX Syntax and Grammar Rules

The following topics describe syntax and grammar rules for MDX functions:

- [“Understanding BNF Notation” on page 933](#)
- [“MDX Grammar Rules” on page 934](#)
- [“MDX Syntax for Specifying Duplicate Member Names and Aliases” on page 947](#)
- [“MDX Axis Specifications” on page 949](#)
- [“MDX Slicer Specification” on page 952](#)
- [“MDX Cube Specification” on page 952](#)
- [“MDX Set Specification” on page 953](#)

- “MDX With Section” on page 954
- “MDX Dimension Specification” on page 958
- “MDX Layer Specification” on page 959
- “MDX Member Specification” on page 960
- “MDX Hierarchy Specification” on page 961
- “MDX Tuple Specification” on page 961
- “MDX Create Set / Delete Set” on page 962

Understanding BNF Notation

This section briefly explains the meaning of symbolic notations used to describe grammar in this document. The query grammar rules are presented using Backus-Naur Form (BNF) syntax notation.

The following table of conventions is not a complete description of BNF, but it can help you read the grammar rules presented in this document.

Symbol	Description	Example
<word> (A word in angle brackets.)	The word presented in angle brackets is not meant to be literally used in a statement; its rules are further defined elsewhere.	When reading the following syntax, <pre>SELECT <axis-specification> ...</pre> you know that <i>axis-specification</i> is not meant to be typed literally into the statement. The rules for <i>axis-specification</i> are further defined in the documentation (look for <i><axis-specification> ::=</i> to get the definition).
<word> ::= (A word in angle brackets, followed directly by the symbol ::=)	A definition, or BNF "production." The symbol ::= can be interpreted to mean "is defined as." The word referred to elsewhere as the placeholder <word> is defined here, directly following <word> ::= .	The following syntax tells you that a <i>tuple</i> is defined as either one member in parenthesis, or two or more comma-separated members in parenthesis. <pre><tuple> ::= ' (' <member> [, <member>] ... ') '</pre>
 The pipe symbol or "OR" symbol.	Precedes alternatives. The symbol can be interpreted to mean "or."	The following syntax: <pre>ON COLUMNS ROWS PAGES CHAPTERS SECTIONS</pre> can be used to build any of the following literal statement parts: <ul style="list-style-type: none"> ● ON COLUMNS ● ON ROWS ● ON PAGES ● ON CHAPTERS ● ON SECTIONS

Symbol	Description	Example
WORD (Text in all caps.)	A query-grammar keyword, to be typed literally.	When reading the following syntax, SELECT <axis-specification> ... you know that SELECT is a keyword, and therefore should be typed literally into its proper location in the statement.
[<word>] or [word] or [WORD] (Square brackets enclosing some word or item.)	An optional element.	In the following high-level query syntax, [<with_section>] SELECT [<axis_specification> [, <axis_specification>...]] FROM [<cube_specification>] [WHERE [< slicer_specification>]] everything, technically, is optional except for SELECT and FROM. Therefore, a query containing only the words SELECT FROM would in fact be valid; however, it would select one consolidated data value from its best estimate of a cube context, which might not be very useful.
[, <word>...] (A comma, a word, and an ellipsis, all enclosed in square brackets.)	You can optionally append a comma-separated list of one or more <words>.	The following syntax SELECT [<axis_specification> [, <axis_specification>...]] indicates that multiple, comma-separated axis specifications can optionally be supplied to the SELECT statement.

MDX Grammar Rules

The following is a comprehensive view of the syntax for MDX in Essbase.

In this document, the syntax for MDX is illustrated using [BNF notation](#).

```

[<with_section>]
SELECT [<axis_specification>
      [, <axis_specification>...]]
[FROM [<cube_specification>]]
[WHERE [<slicer_specification> [<dim_props>]]

<cube_specification> ::=
    '[' <ident_or_string>.<ident_or_string> ']'
    | <delim_ident>.<delim_ident>

<delim_ident> ::=
    '[' <ident> ']'
    | <ident_or_string>

<ident_or_string> ::=
    '<ident>'
    | <ident>

```

Note: <ident> refers to a valid Essbase application/database name. In the cube specification, if there are two identifiers, the first one should be application name and the second one should be database name. For example, all of the following are valid identifiers:

- Sample.Basic
- [Sample.Basic]
- [Sample].[Basic]
- 'Sample'. 'Basic'

```
<axis_specification> ::=  
    [NON EMPTY] <set> [<dim_props>] ON  
    COLUMNS | ROWS | PAGES | CHAPTERS |  
    SECTIONS | AXIS (<unsigned_integer>)
```

```
<dim_props> ::=  
    [DIMENSION] PROPERTIES <property> [, <property>...]
```

```
< slicer_specification > ::= <set> | <tuple> | <member>
```

Note: The cardinality of the <set> in the slicer should be 1.

```
<member> ::=  
    <member-name-specification>  
    | <member_value_expression>
```

```
<member-name-specification> ::=
```

A **member name** can be specified in the following ways:

1. By specifying the actual name or the alias; for example, Cola, Actual, COGS, and [100].

If the member name starts with number or contains spaces, it should be within braces; for example, [100]. Braces are recommended for all member names, for clarity and code readability.

For attribute members, the long name (qualified to uniquely identify the member) should be used; for example, [Ounces_12] instead of just [12].

2. By specifying dimension name or any one of the ancestor member names as a prefix to the member name; for example, [Product].[100-10] and [Diet].[100-10] This is a recommended practice for all member names, as it eliminates ambiguity and enables you to refer accurately to shared members.

Note: Use only one ancestor in the qualification. Essbase returns an error if multiple ancestors are included. For example, [Market].[New York] is a valid name for New York, and so is [East].[New York]. However, [Market].[East].[New York] returns an error.

3. By specifying the name of a calculated member defined in the WITH section.

4. For outlines that have duplicate member names enabled, see also [“MDX Syntax for Specifying Duplicate Member Names and Aliases”](#) on page 947.

```

<member_value_expression> ::=
    Parent ( <member> [, <hierarchy>] )
        | <member>.Parent [( <hierarchy> ) ]
    | FirstChild ( <member> )
        | <member>.FirstChild
    | LastChild ( <member> )
        | <member>.LastChild
    | PrevMember ( <member> [, <layertype>] )
        | <member>.PrevMember [( <layertype> ) ]
    | NextMember ( <member> [, <layertype>] )
        | <member>.NextMember [( <layertype> ) ]
    | FirstSibling ( <member> [, <hierarchy>] )
        | <member>.FirstSibling [( <hierarchy> ) ]
    | LastSibling ( <member> [, <hierarchy>] )
        | <member>.LastSibling [( <hierarchy> ) ]
    | Ancestor ( <member> , <layer> | <index> [, <hierarchy>] )
    | Lead ( <member> , <index> [, <layertype>] [, <hierarchy>] )
        | <member>.Lead ( <index> [, <layertype>] [, <hierarchy>] )
    | Lag ( <member> , <index> [, <layertype>] [, <hierarchy>] )
        | <member>.Lag ( <index> [, <layertype>] [, <hierarchy>] )
    | CurrentMember ( <dim_hier> )
        | <dim_hier>. CurrentMember
    | DefaultMember ( <dim_hier> )
        | <dim_hier>. DefaultMember
    | OpeningPeriod ( [<layer> [, <member>]] )
    | ClosingPeriod ( [<layer> [, <member>]] )
    | Cousin ( <member> , <member> )
    | ParallelPeriod ( [<layer> [, <index> [, <member> [, <hierarchy>]]]] )
    | Item ( <tuple> , <index> )
        | tuple[.Item] ( <index> )
    | LinkMember ( <member> , <hierarchy> )
        | member.LinkMember ( <hierarchy> )
    | DateToMember ( <date> , <dim_hier> [ , <genlev> ] )
    | StrToMbr ( <string_value_expr> [, <dimension>] [, MEMBER_NAMEONLY |
<alias_table_name>] )

```

```

<dim_hier> ::= <dimension>

```

```

<dimension> ::= =
    <dimension-name-specification>
    | Dimension ( <member> | <layer> )
        | <member>.DIMENSION
        | <layer>.DIMENSION

```

```

<dimension-name-specification> ::=
    Same as <member_name-specification> case 1.
    e.g. Product, [Product]

```

```

<hierarchy> ::=

```

A **hierarchy** refers to a root member of an alternate hierarchy, which is always at generation 2 of a dimension. Member value expressions are not allowed as hierarchy arguments.


```

<layertype> ::=
    GENERATION | LEVEL

<layer> ::=
    <layer-name-specification>
    | Levels ( <dim_hier>, <index> )
      | <dim_hier>.Levels ( <index> )
    | Generations ( <dim_hier>, <index> )
      | <dim_hier>.Generations ( <index> )
    | <member>.Generation
    | <member>.Level

<layer-name-specification> ::=

```

A **layer name** can be specified in the following ways:

1. By specifying the generation or level names; for example, `States` or `Regions`.

The generation or level name can be within braces; for example, `[Regions]`. Using braces is recommended.

2. By specifying the dimension name along with the generation or level name; for example, `Market.Regions` and `[Market].[States]` This naming convention is recommended.

```

<tuple> ::=
    <member>
    | ( <member> [, <member>].. )
    | <tuple_value_expression>

```

A **tuple** is a collection of member(s) with the restriction that no two members can be from the same dimension. For example, `(Actual, Sales)` is a tuple. `(Actual, Budget)` is not a tuple, as both members are from the same dimension.

```

<tuple_value_expression> ::=
    CurrentTuple ( <set> )
      | <set>.Current
    | Item ( <set>, <index> )
      | <set>[.Item] (<index>)

```

A **set** is a collection of tuples where members in all tuples must be from the same dimensions and in the same order.

For example, `{(Actual, Sales), (Budget, COGS)}` is a set.

`{(Actual, Sales), (COGS, [100])}` is not a set because the second tuple has members from Scenario and Product dimensions, whereas the first tuple has members from Scenario and Measures dimensions.

`{(Actual, Sales). (COGS, Budget)}` is not a set because the second tuple has members from Scenario and Measures dimensions, whereas the first tuple has members from Measures and Scenario dimensions (the order of dimensions is different).

Note: The size of an input set to a function has range between 0 and 4294967295 tuples.

```

<set> ::=
    MemberRange ( <member>, <member>
                  [, <layertype>] [, <hierarchy>] )
    | <member> : <member>
    | { <tuple> | <set> [, <tuple> | <set>] .. }
    | ( <set> )
    | <set_value_expression>

<set_value_expression> ::=
    | Members ( <dim_hier> )
      | <dim_hier>.Members
    | Members ( <layer> )
      | <layer>.Members
    | Children ( <member> )
      | <member>.Children
    | CrossJoin ( <set> , <set> )
    | Union ( <set> , <set> [, ALL] )
    | Intersect ( <set> , <set> [, ALL] )
    | Except ( <set> , <set> [, ALL] )
    | Extract ( <set> , <dim_hier> [, <dim_hier>] .. )
    | Head ( <set> [, <index>] )
    | Subset ( <set> , <index> [, index] )
    | Tail ( <set> [, index] )
    | Distinct ( <set> )
    | Siblings ( <member> [, <selection_flags>, [INCLUDEMEMBER|EXCLUDEMEMBER]] )
      | <member>.Siblings
    | Descendants ( <member> , [{<layer>|<index>}[, <Desc_flags>]] )
    | PeriodsToDate ( [<layer>[, <member> [, <hierarchy>]]] )
    | LastPeriods ( <index>[, <member> [, <hierarchy>]] )
    | xTD ( [<member>] )
      where xTD could be {HTD|YTD|STD|PTD|QTD|MTD|WTD|DTD}
    | Hierarchize ( <set> [, POST] )
    | Filter ( <set> , <search_condition> )
    | Order ( <set>, <value_expression> [, BASC | BDESC] )
    | TopCount ( <set> , <index> [, <numeric_value_expression>] )
    | BottomCount ( <set> , <index> [, <numeric_value_expression>] )
    | TopSum ( <set> , <numeric_value_expression>
              , <numeric_value_expression> )
    | BottomSum ( <set> , <numeric_value_expression>
                 , <numeric_value_expression> )
    | TopPercent ( <set> , <percentage> , <numeric_value_expression> )
    | BottomPercent ( <set> , <percentage> , <numeric_value_expression> )
    | Generate ( <set> , <set> [, [ALL]] )
    | DrilldownMember ( <set> , <set>[, RECURSIVE] )
    | DrillupMember ( <set> , <set> )
    | DrilldownByLayer ( <set> [, {<layer>|<index>} )
      | DrilldownLevel ( <set> [, {<layer>|<index>} )
    | DrillupByLayer ( <set> [, <layer>] )
      | DrillupLevel ( <set>[, <layer>] )
    | WithAttr ( <member> , <character_string_literal>, <value_expression> )
    | WithAttrEx ( <member> , <character_string_literal>, <value_expression>, ANY,
<tuple> | <member> [, <tuple> | <member>] )
    | Attribute ( <member> )
    | AttributeEx ( <member>, ANY, <tuple> | <member> [, <tuple> | <member>] )
    | Uda ( <dimension> | <member> , <string_value_expression> )
    | RelMemberRange ( <member>, <prevcount>, <nextcount>,

```

```

    [, <layertype>] [, <hierarchy>] )
| Ancestors ( <member>, <layer> | <index> )
| <conditional_expression>

```

Note: <conditional_expression> is expected to return a <set> in the above production.

```

<Desc_flags> ::=
    SELF
    | AFTER
    | BEFORE
    | BEFORE_AND_AFTER
    | SELF_AND_AFTER
    | SELF_AND_BEFORE
    | SELF_BEFORE_AFTER
    | LEAVES

<selection_flags> ::=
    LEFT
    | RIGHT
    | ALL

<value_expression> ::=
    <numeric_value_expression>
    | <string_value_expression>

<numeric_value_expression> ::=
    <term>
    | <numeric_value_expression> + <term>
    | <numeric_value_expression> - <term>

<term> ::=
    <factor>
    | <term> * <factor>
    | <term> / <factor>

<factor> ::=
    [+ | -] <numeric_primary>

<numeric_primary> ::=
    <value_expr_primary>
    | <numeric_value_function>
    | <mathematical_function>
    | <date_function>

```

Note: The data type of <value_expr_primary> in the above production must be numeric.

```

<base> ::=
    <numeric_value_expression>

<power> ::=
    <numeric_value_expression>

<mathematical_function> ::=
    Abs ( <numeric_value_expression> )
    | Exp ( <numeric_value_expression> )

```

```

| Factorial ( <index> )
| Int ( <numeric_value_expression> )
| Ln ( <numeric_value_expression> )
| Log ( <numeric_value_expression> [, <base>] )
| Log10 ( <numeric_value_expression> )
| Mod ( <numeric_value_expression> , <numeric_value_expression> )
| Power ( <numeric_value_expression> , <power> )
| Remainder ( <numeric_value_expression> )
| Stddev ( <set> [, <numeric_value_expression> [, IncludeEmpty] ] )
| Stddevp ( <set> [, <numeric_value_expression> [, IncludeEmpty] ] )
| Round ( <numeric_value_expression> , <index> )
| Truncate ( <numeric_value_expression> )

```

```

<date_function> ::=
    DateRoll(<date>, <date_part>, <index>)
| DateDiff(<date>, <date>, <date_part>)
| DatePart(<date>, <date_part>)
| Today()
| TodateEx(<date_format_string>, <string>)
| GetFirstDate(<member>)
| GetLastDate(<member>)
| UnixDate(<numeric_value_expression>)
| GetFirstDay(<date>, <date_part>)
| GetLastDay(<date>, <date_part>)
| GetNextDay(<date>, <week-day-specification>, [0|1] )
| GetRoundDate(<date>, <date_part>)

```

The <date> argument is a number representing the input date. The number is the number of seconds elapsed since midnight, January 1, 1970. To retrieve this number, use any of the following functions: Today(), TodateEx(), GetFirstDate(), GetLastDate().

```

<date_part> ::=
    DP_YEAR
| DP_QUARTER
| DP_MONTH
| DP_WEEK
| DP_DAY
| DP_DAYOFYEAR
| DP_WEEKDAY

```

Note: DP_DAYOFYEAR and DP_WEEKDAY are not valid arguments in functions DateRoll and DateDiff.

```

<week-day-specification> ::=
    1 | 2 | 3 | 4 | 5 | 6 | 7
    e.g. 1 implying Sunday, 7 implying Saturday

```

```

<date_format_string> ::=
    "mon dd yyyy"
| "Month dd yyyy"
| "mm/dd/yy"
| "mm/dd/yyyy"

```

```

| "yy.mm.dd"
| "dd/mm/yy"
| "dd.mm.yy"
| "dd-mm-yy"
| "dd Month yy"
| "dd mon yy"
| "Month dd, yy"
| "mon dd, yy"
| "mm-dd-yy"
| "yy/mm/dd"
| "yymmdd"
| "dd Month yyyy"
| "dd mon yyyy"
| "yyyy-mm-dd"
| "yyyy/mm/dd"
| "Long format"
| "Short format"

```

```

<string_value_expression> ::=
    <string_value_primary>
    | FormatDate (<date>, <date_format_string>)
    | Concat (<string_value_expression> [<, string_value_expression> ...])
    | Left (<string_value_expression>, <length>)
    | Right (<string_value_expression>, <length>)
    | Substring (<string_value_expression>, <index> [, <index>])
    | Upper (<string_value_expression>)
    | Lower (<string_value_expression>)
    | RTrim (<string_value_expression>)
    | LTrim (<string_value_expression>)
    | NumToStr (<value_expr_primary>)
    | EnumText (<textlistname> | <member>, <numeric_value_expression>)

```

```

<value_expr_primary> ::=
    <unsigned_numeric_literal>
    | ( <numeric_value_expression> )
    | <tuple> [.RealValue]
    | <member> [.RealValue]
    | <tuple> [.Value]
    | <member> [.Value]
    | CellValue()
    | <property>
    | <conditional_expression>
    | MISSING

```

```

<string_value_primary> ::=
    <character_string_literal>
    | <string_property>

```

Notes

- <conditional_expression> is expected to return a numeric value in the above production.
- String literals are delimited by double quotes("").

```

<conditional_expression> ::=
    <if_expression>

```

```

| <case_expression>
| CoalesceEmpty ( <numeric_value_expression>
                  , <numeric_value_expression>)

<case_expression> ::=
    <simple_case> | <searched_case>

<if_expression> ::=
    IIF ( <search_condition>, <>true_part>, <>false_part> )
<true_part> ::=
    <value_expression> | <set>
<>false_part> ::=
    <value_expression> | <set>

<simple_case> ::=
    Case <case_operand>
        <simple_when_clause>...
        [ <else_clause> ]
    END

<simple_when_clause> ::=
    WHEN <when_operand>
        THEN <result>
<else_clause> ::=
    ELSE <value_expression> | <set>

<case_operand> ::=
    <value_expression>
<when_operand> ::=
    <value_expression>
<result> ::=
    <value_expression> | <set>

<searched_case> ::=
    Case
        <searched_when_clause>...
        [ <else_clause> ]
    END

<searched_when_clause> ::=
    WHEN <search_condition>
        THEN <result>

<numeric_value_function> ::=
    Avg ( <set> [, <numeric_value_expression>] [, IncludeEmpty] )
| Max ( <set> [, <numeric_value_expression>])
| Min ( <set> [, <numeric_value_expression>])
| Sum ( <set> [, <numeric_value_expression>])
| NonEmptyCount ( <set> [, <numeric_value_expression>])
| Count ( <set> [, IncludeEmpty] )
| <dts-specification> ::= DTS (<dts-operation-specification>,<member>)
   <dts-operation-specification> ::= HTD|YTD|STD|PTD|QTD|MTD|WTD|DTD
| Todate ( <string_value_expression> , <string_value_expression> )
| Ordinal (<layer>)
| Aggregate (<set> [,<member-name-specification>])
| Rank (<member_or_tuple>, <set> [,<numeric_value_expression>
    [, <rank_flags>]])

```

```

| NTile (<member_or_tuple>, <set>, <index>,
        <numeric_value_expression>)
| Percentile (<set>, <numeric_value_expression>,
             <numeric_value_expression>)
| Median (<set>, <numeric_value_expression>)
| Len (<string_value_expression>)
| InStr (<index>, <string_value_expression>,
        <string_value_expression>, <numeric_value_expression>)
| StrToNum (<string_value_expression>)
| EnumValue(<enum_string>)
| JulianDate(<date>)

```

Note: The <member-name-specification> in Aggregate function should refer to an Accounts dimension member name.

Note: <enum_string> represents an enumerated string. It should be in the following format. The member should refer to a member of type text.

```

<enum_string> ::=
    <textlist-name-specification>.<character_string_literal>
    | <member>.<character_string_literal>
<textlist-name-specification> ::=
    Same as <member_name-specification> case 1. The text list name specification should
    refer to the name of a text list object.
    e.g. AccountStatus, [AccountStatus]

<member_or_tuple> ::=
    <member>
    | <tuple>

<index> ::=
    <numeric_value_expression>

```

Note: The input <index> argument has range between -2147483647 and 2147483647.

```

<percentage> ::=
    <numeric_value_expression>

<search_condition> ::=
    <bool_term>
    | <search_condition> OR <bool_term>

<bool_term> ::=
    <bool_factor>
    | <bool_term> AND <bool_factor>

<bool_factor> ::=
    <bool_primary>
    | NOT <bool_primary>

<bool_primary> ::=
    <value_expression> [=|>|<|<>|>=|<=] <value_expression>

```

```

| <property> IN <member>|<character_string_literal>
| <property>
| IsEmpty ( <value_expression> )
| ( <search_condition> )
| IsSibling(<member>,<member> [, INCLUDEMEMBER])
| IsLeaf(<member>)
| IsGeneration(<member>,<index>)
| IsLevel(<member>,<index>)
| IsAncestor(<member>,<member> [, INCLUDEMEMBER])
| IsChild(<member>,<member> [, INCLUDEMEMBER])
| IsUda (<member>, <string_value_expression>)
| IsAcctType (<member>, <AcctTag>)
| Is ( <member> , <member> )
|   | <member> Is <member>
| IsValid (<member> | <tuple> | <set> | <layer> | <property>)
| IsMatch (<string_value_expression>, <string_value_expression>, [,MATCH_CASE|
IGNORE_CASE])
| Contains (<member_or_tuple>, <set>)

```

Note: Only properties with boolean values can be used as <bool_primary>.

```

<AcctTag> ::=
    FIRST
    | LAST
    | AVERAGE
    | EXPENSE
    | TWO-PASS

<rank_flags> ::=
    ORDINALRANK
    | DENSERANK
    | PERCENTRANK

<with_section> ::=
    WITH <frml_spec>

<frml_spec> ::=
    <single_frml_spec>
    | <frml_spec> <single_frml_spec>

<single_frml_spec> ::=
    <set_spec>
    | <perspective_specification>
    | <member_specification>

<set_spec> ::=
    SET <set_name> AS ' <set> '

<set_name> ::=

```

The name of the set to be defined. The name cannot be same as any names/aliases of database members, generation/level names, or UDA names.

```

<perspective_specification> ::=

```



```

PERSPECTIVE REALITY | <tuple> FOR <dimension-name-specification>

<member_specification> ::=
    MEMBER <member_name> AS '
        <nonempty_specification>
        <numeric_value_expression> '
    [, <solve_order_specification>]

<member_name> ::=
    <dimension-name-specification>.<calculated member name>

<calculated member name> ::=

```

Names used for calculated members cannot be the same as any names/aliases of database members, generation/level names, or UDA names.

```

<solve_order_specification> ::=
    SOLVE_ORDER = <unsigned_integer>

<property> ::=
    <member>.<property_specification>
    | <dim_hier>.<property_specification>
    | <property_specification>

```

Note: The last two alternatives in the above rule can be used only inside the DIMENSION PROPERTIES section.

For example, assume an axis has 2 dimensions, Product and Market. Using DIMENSION PROPERTIES Gen_number, [Product].level_number, the generation number will be present in the output for the members of both dimensions, whereas the level number will be present only for the members of the Product dimension.

Within a value expression, [Product].Gen_number refers to the generation number of the member named [Product].

[Product].CurrentMember.Gen_number refers to the generation number of the current member of the [Product] dimension.

For example,

```
Filter ([Product].Members, [Product].Gen_number > 1)
```

returns an empty set. Product.Generation is 1, so the search condition fails for each tuple of [Product].Members.

```
Filter ([Product].Members, [Product].CurrentMember.Gen_number > 1)
```

returns all members of Product dimension except the top dimension member, [Product].

```
<string_property> ::= <member>.<property_specification>
```

Note: The above rule specifies string properties such as MEMBER_NAME, MEMBER_ALIAS.

```

<property_specification> ::=
    MEMBER_NAME
    | MEMBER_ALIAS
    | GEN_NUMBER
    | LEVEL_NUMBER
    | <dimension-name-specification>
    | <uda-specification>

```

Note: The <dimension-name-specification> in <property_specification> should be an attribute dimension-name specification. The attribute dimension names are treated as properties of members from their corresponding base dimensions.

```

<uda-specification> ::=

```

The <uda-specification> specifies a User Defined Attribute(UDA). UDA properties are Boolean-valued properties. A TRUE value indicates presence of a UDA for a member. For example,

```

Filter (Market.Members, Market.CurrentMember.[Major Market])

```

returns the Market dimension members tagged with "Major Market" UDA in the outline.

For more discussion of properties, see [“About MDX Properties” on page 964](#).

The following rule describes the syntax for **Essbase outline formulas** in aggregate storage applications.

```

<formula_specification> ::= <nonempty_specification>
                           <numeric_value_expression>

```

```

<nonempty_specification> ::= NONEMPTYMEMBER <nonempty_member_list>
                             | NONEMPTYTUPLE ( <nonempty_member_list> )

```

```

<nonempty_member_list> ::= <nonempty_member_name>
                          | <nonempty_member_name> [ , <nonempty_member_list> ]

```

```

<nonempty_member_name> ::=
    An Essbase member name or a calculated member name (only when used in another
    calculated member).

```

Note: The member name (or member names when multiple names are specified) in a NONEMPTYMEMBER directive should belong to the same dimension as the calculated member or formula member in which it is specified.

```

<signed_numeric_literal> ::=
    [+|-] <unsigned_numeric_literal>

```

```

<unsigned_numeric_literal> ::=
    <exact_numeric_literal>
    | <approximate_numeric_literal>

```

```

<exact_numeric_literal> ::=
    <unsigned_integer>[.<unsigned_integer>]

```

```

| <unsigned_integer>.
| .<unsigned_integer>

<unsigned_integer> ::=
    {<digit>}...

<approximate_numeric_literal> ::=
    <mantissa>E<exponent>

<mantissa> ::=
    < exact_numeric_literal>

<exponent> ::=
    [<sign>]<unsigned_integer>

<digit> ::=
    0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

```

Note: Numbers can also be input in scientific notation (mantissa/exponent), using the E character.

```

<character_string_literal> ::=
    <quote>[<character_representation>...] <quote>

<character_representation> ::=
    <nonquote_character>
    | <quote_symbol>

<nonquote_character> ::=
    Any character in the character set other than <quote>

<quote_symbol> ::=
    <quote> <quote>

<quote> ::= "

```

The following is the syntax for Format Strings in Essbase:
MdxFormat(string_value_expression)

MDX Syntax for Specifying Duplicate Member Names and Aliases

The following member specification rules apply to databases with duplicate member names enabled.

Qualified names must be used to specify duplicate member names. Qualified member or alias names can be specified using:

- **Fully qualified member names**—Consist of duplicate member or alias name and all ancestors up to and including the dimension name. Each name must be enclosed in square brackets([]) and separated by a period.

[DimensionMember].[Ancestors...].[DuplicateMember]

For example:

[Product].[100].[100-10]

- **Shortcut qualified member names**—Essbase internally constructs shortcut qualified names for members in duplicate member outlines. These can be inserted into scripts using Administration Services by right clicking on the member and selecting Insert member name. You can also manually insert shortcut qualified names into scripts, spreadsheets, or MDX queries.

Essbase uses the following syntax to construct shortcut qualified names. Using the same syntax that Essbase uses when you reference members in scripts, spreadsheets, and MDX queries is optimal, but not required.

Scenario	Qualified Name Syntax	Example
Duplicate member names exist at generation 2	[DimensionMember].[DuplicateMember]	[Year].[Jan] or [Product].[Jan]
Duplicate member names exist in an outline, but are unique within a dimension	[DimensionMember]@[DuplicateMember]	[Year]@[Jan]
Duplicate member names have a unique parent	[ParentMember].[DuplicateMember]	[East].[New York]
Duplicate member names exist at generation 3	[DimensionMember].[ParentMember].[DuplicateMember]	[Products].[Personal Electronics]. [Televisions]
Duplicate member names exist at a named generation or level, and the member is unique at its generation or level	[DimensionMember]@[GenLevelName] [DuplicateMember]	[2006]@[Gen1] [Jan]

In MDX, either one the following syntax methods must be used to reference shortcut qualified member names:

- **Escape Character method**—Because MDX syntax also uses square brackets:
 1. Any internal closing bracket (]) used by name parts within the shortcut qualified names requires an additional] escape character.
 2. The entire shortcut qualified member name must be enclosed in a set of square brackets ([]).

Examples:

[Year].[Jan] is referenced as [[Year]].[Jan]] in MDX.

[Year]@[Jan] is referenced as [[Year]]@[Jan]] in MDX.

[2006]@[Gen1] | [Jan] is referenced as [[2006]]@[Gen1]] | [Jan]] in MDX.

Note: The above syntax also works for fully qualified member names, but is not required.

- **StrToMbr Function method**—You can use the StrToMbr function to convert qualified name strings to member value expressions.

Examples:

[Year] . [Jan] is referenced as StrToMbr (" [Year] . [Jan] ") in MDX.

[Year]@[Jan] is referenced as StrToMbr (" [Year]@[Jan] ") in MDX.

[2006]@[Gen1] | [Jan] is referenced as StrToMbr (" [2006]@[Gen1] | [Jan] ") in MDX.

Note: The above syntax also works for fully qualified member names, but is not required.

Query Example

The following query uses both methods of referencing shortcut member names in MDX:

```
SELECT
  { Sales, Profit }
ON COLUMNS,
  {[Store]]@[6]], StrToMbr("Product.SKU.1")}
ON ROWS
FROM MySample.Basic
WHERE ([[1998]].[Q1]].[1]])
```

Note: StrToMbr accepts any type of member-identifier strings: names, aliases or qualified names.

MDX Axis Specifications

An axis specification consists of a set and one or more axis keywords.

```
<axis_specification> ::= =
  [NON EMPTY] <set> ON COLUMNS|ROWS|PAGES|CHAPTERS|SECTIONS|AXIS(<unsigned_integer>)
```

Understanding the following concepts will help you construct axis specifications for many SELECT queries

Ordering of Axes

If providing multiple axes, you cannot skip axes. For example, you can specify a Row axis only if you have a Column axis. You can specify a Pages axis only if you also have Column and Row axes.

You can also use ordinals to represent the axes. For example, you can specify <set> ON AXIS(0), <set> ON AXIS(1), etc.

You can specify up to 64 axes (though it is common to use just two). The first five ordinal axes have keyword aliases:

Axis Keyword	Axis Ordinal
COLUMNS	AXIS(0) (default if nothing specified)
ROWS	AXIS(1)
PAGES	AXIS(2)
CHAPTERS	AXIS(3)
SECTIONS	AXIS(4)

For example:

```
SELECT set1 ON COLUMNS,
set2 ON ROWS
FROM Sample.Basic
```

is the same as:

```
SELECT set1 ON AXIS(0),
set2 ON AXIS(1)
FROM Sample.Basic
```

Both return a hypothetical data cube (or subset) of the following format:

(axis)	Member names in set1
Member names in set2	Data at intersections of set1 and set2 members

The examples above are hypothetical because they will not return a cube until values are provided for the sets. In the following example, we replace set1 and set2 with real sets:

```
SELECT
{[100-10], [100-20]} ON COLUMNS,
{[Qtr1], [Qtr2], [Qtr3], [Qtr4]} ON ROWS
FROM Sample.Basic
```

which returns the following results:

(axis)	100-10	100-20
Qtr1	5096	1359
Qtr2	5892	1534
Qtr3	6583	1528
Qtr4	5206	1287

Specifying the Set

You can represent the sets in each axis in many ways.

```
SELECT
{ }
ON COLUMNS
from sample.basic
```

illustrates that you can choose nothing for a set. However, no cell values will be returned. The following rules apply:

- When any of the axes contains an empty set, no cell values are returned. The axes whose sets have at least one tuple will have their tuples returned.
- If there are no axes at all, then exactly one cell is returned using the default member of each dimension. The slicer tuple, if present, overrides the default member for the respective dimensions.

```
SELECT
{ ( [Year].[Qtr2] ) }
ON COLUMNS
from sample.basic
```

illustrates using a set that contains a single tuple.

For more information about sets, see [“MDX Set Specification” on page 953](#).

NON EMPTY

The axis specification syntax including NON EMPTY is shown below:

```
<axis_specification> ::=
    [NON EMPTY] <set> ON
    COLUMNS | ROWS | PAGES | CHAPTERS |
    SECTIONS | AXIS (<unsigned_integer>)
```

Including the optional keywords NON EMPTY before the set specification in an axis causes suppression of slices in that axis that would contain entirely #MISSING values.

For any given tuple on an axis (such as (Qtr1, Actual)), a slice consists of the cells arising from combining this tuple with all tuples of all other axes. If all of these cell values are #MISSING, the NON EMPTY keyword causes the tuple to be eliminated.

For example, if even one value in a row is not empty, the entire row is returned. Including NON EMPTY at the beginning of the row axis specification would eliminate the following row slice from the set returned by a query:

Qtr1					
Actual	#Missing	#Missing	#Missing	#Missing	#Missing

For another example, see the [Tail](#) function.

Dimension Properties

A property, in MDX grammar, refers to the Essbase concepts of attributes and UDAs.

The axis specification syntax including the properties specification is shown below:

```
<axis_specification> ::=  
    [NON EMPTY] <set> [<dim_props>] ON  
    COLUMNS | ROWS | PAGES | CHAPTERS |  
    SECTIONS | AXIS (<unsigned_integer>)
```

As shown in the above syntax, a properties specification can follow the set specification in an axis.

For more information about properties, see [“About MDX Properties” on page 964](#).

MDX Slicer Specification

This section shows rules for the slicer specification (WHERE clause). The slicer axis is a way of limiting a query to apply only to a specific area of the database.

A slicer specification consists of the WHERE keyword followed by a tuple, member, or set. You can optionally query for certain dimension properties in the slicer specification.

Syntax

```
[WHERE [<slicer_specification>] [<dim_props>]]  
  
<slicer_specification> ::= <set> | <tuple> | <member>
```

Note: The cardinality of the <set> in the slicer should be 1; in other words, if a set is used, it must evaluate to a single tuple.

```
<dim_props> ::=  
    [DIMENSION] PROPERTIES <property> [, <property>...]
```

Example

For example, you may want an entire query to apply only to Actual Sales in the Sample Basic database, excluding budgeted sales or any other measures. The WHERE clause might look like the following:

```
SELECT  
    {[West].children}  
ON COLUMNS,  
    {[Diet].children}  
ON ROWS  
FROM Sample.Basic  
WHERE ([Scenario].[Actual], [Measures].[Sales])
```

MDX Cube Specification

Use the cube specification to name the database at which the query is directed. A cube specification consists of the FROM keyword followed by delimited or nondelimited identifiers indicating an application name and a database name.

The first identifier should be an application name and the second one should be a database name. For example, all of the following are valid identifiers:

- `Sample.Basic`
- `[Sample.Basic]`
- `[Sample].[Basic]`
- `'Sample'.'Basic'`

Syntax

```
[FROM [<cube_specification>]]

<cube_specification> ::=
    '['<ident_or_string>.<ident_or_string>']'
    |<delim_ident>.<delim_ident>

<delim_ident> ::=
    '[' <ident> ']'
    |<ident_or_string>

<ident_or_string> ::=
    '<ident>'
    |<ident>
```

Notes

If `[FROM [<cube_specification>]]` is omitted from a query, the current database context is assumed.

Example

`Sample.Basic` is the cube specification in the following hypothetical query.

```
SELECT
...
FROM Sample.Basic
```

MDX Set Specification

A set is a collection of [tuples](#). In each tuple of the set, members must represent the same dimensions as do the members of other tuples of the set. Additionally, the dimensions must be represented in the same order.

```
<set> ::=
    MemberRange ( <member>, <member> )
    | <member> : <member>
    | { [<tuple> | <set>] [, <tuple> | <set>].. }
    | <set_value_expression>
```

Item	Description
MemberRange (<member>, <member>)	A set can be a range of members, specified using the MemberRange function.
<member> : <member>	Alternate syntax that has the same effect as the MemberRange function.
{[<tuple> <set>] [, <tuple> <set>]. · } · }	Unless it is returned by a function, a set must be enclosed in curly braces { }. A set can be one or more tuples , or it can be made up of other sets. All tuples in a set must have the same dimensionality.
<set_value_ expression>	Output from any function that returns a set. As an alternative to creating sets member-by-member or tuple-by-tuple, you can use a function that returns a set. For a list of functions that return sets, see "MDX Functions" on page 989 .

MDX With Section

The WITH section is for defining referential sets or members that can be used multiple times during the life of a query.

Beginning with the keyword `WITH` at the very start of a query, you can define a buffer of reusable logic lasting for the length of the query execution. This can save time in lines of code written as well as in execution time.

If varying attributes are enabled, the WITH section can also be used to define perspective for each varying attribute dimension. In case of multiple varying attributes, perspective setting can be defined for each varying attribute dimension separately.

In the WITH section, you can create the following reusable elements:

- Calculated members
- Named Sets

Syntax

WITH

```
SET set_name AS ' set '
| MEMBER calculated_member_name AS ' <numeric_value_expr> '
| [, <solve_order_specification> ]
| <perspective_specification>
```

Item	Description
<p><i>set_name</i></p>	<p>The name of the set that will be defined after the <code>AS</code> keyword. Any name can be used; it should be something that helps you remember the nature of the set. For example, a set name could be <code>Best5Books</code>, which names a set of the five top-selling paperback titles in December:</p> <pre>WITH SET [Best5Books] AS 'Topcount ([Paperbacks].members, 5, ([Measures].[Sales], [Scenario].[Actual], [Year].[Dec]))'</pre>
<p><i>set</i></p>	<p>The logic of a set specification; this can be re-used because it is being named. Must be enclosed in single quotation marks. In the example above, the <code>Topcount</code> function defines the entire set.</p>
<p><i>calculated_member_name</i></p>	<p>A name for a hypothetical member existing for the duration of query execution. In its definition, you must associate the calculated member with a dimension (as <code>[Max Qtr2 Sales]</code> is associated with the <code>Measures</code> dimension, in the example that follows).</p> <p>For example, the calculated member named <code>Max Qtr2 Sales</code> has its value calculated at execution time using the <code>Max</code> function:</p> <pre>WITH MEMBER [Measures].[Max Qtr2 Sales] AS 'Max ({[Year].[Qtr2]}, [Measures].[Sales])'</pre> <p>Calculated members do not work with metadata functions such as <code>Children</code>, <code>Descendants</code>, <code>Parent</code>, and <code>Siblings</code>. For example, if there is a calculated member defined as <code>[CM1]</code>, you cannot use it in the following way: <code>[CM1].children</code>.</p>
<p><numeric_value_expr></p>	<p>An expression involving real members in the database outline, compared using mathematical functions. The value resulting from the expression is applied to the calculated member. By using calculated members, you can create and analyze a great many scenarios without the need to modify the database outline.</p>
<p><solve_order_specification></p>	<p>Optional. By adding <code>, SOLVE_ORDER = n</code> to the end of each calculated member, you can specify the order in which the members are calculated. For example, solve order in the following hypothetical query is indicated in bold:</p> <pre>WITH MEMBER [Product].[mbr1] AS 'calculation', SOLVE_ORDER = 2 MEMBER [Product].[mbr2] AS 'calculation', SOLVE_ORDER = 1 SELECT {[Year].children} on columns, { [Product].[mbr1], [Product].[mbr2] } on rows</pre> <p>See Usage Examples for Solve Order.</p>

Item	Description
<perspective_specification>	<p>PERSPECTIVE REALITY <i>tuple FOR dimension</i></p> <p>When a database uses varying attributes, base members associated with the varying attributes are aggregated according to the specified perspective.</p> <p>You can set the perspective to reality (using the REALITY keyword) or to explicit (using an input tuple consisting of level 0 members).</p> <p>Reality-based evaluation and reporting is the default, in which independent members are determined by the current context.</p> <p>When using explicit evaluation and reporting, you specify a tuple of level 0 members from the independent dimension to be used as the context.</p> <p>For an example of a reality-based perspective, see the example for AttributeEx. For an example of an explicit perspective, see the example for WithAttrEx.</p>

Usage Examples for Solve Order

```
WITH
MEMBER
    [Measures].[Profit Percent]
    AS 'Profit *100 /Sales', SOLVE_ORDER=20
MEMBER
    [Year].[FirstFourMonths]
    AS 'Sum(Jan:Apr)', SOLVE_ORDER=10
SELECT
    {[Profit], [Sales], [Profit Percent]}
ON COLUMNS,
    {[Jan], [Feb], [Mar], [Apr], [FirstFourMonths]}
ON ROWS
FROM Sample.Basic
```

The calculated member [Profit Percent], defined in the Measures dimension, calculates Profit as a percentage of Sales.

The calculated member [FirstFourMonths], defined in the Year dimension, calculates sum of data for first four months.

When data for ([Profit Percent], [FirstFourMonths]) is evaluated, SOLVE_ORDER specifies the order of evaluation, ensuring that [Profit Percent] is evaluated first, and resulting in a correct value for percentage. If you change the order of evaluation, you will see that the percentage value is not correct. In this example, SOLVE_ORDER specifies that sum should be calculated before percentage.

Tie-Case Example for Solve Order

When evaluating a cell identified by multiple calculated members, the SOLVE_ORDER value is used to determine the order in which the expressions are evaluated. The expression that is used to evaluate the cell is that of the calculated member with the highest SOLVE_ORDER value. In this case, [Profit Percent]'s expression is used to evaluate ([Profit Percent], [FirstFourMonths]). The example above is calculated as:

```
([Profit Percent], [FirstFourMonths])
= ([Profit], [FirstFourMonths]) * 100 / ([Sales], [FirstFourMonths])
= (([Profit], [Jan]) + ([Profit], [Feb]) + ([Profit], [Mar]) + ([Profit], [Apr])) *
```

```
(([Sales], [Jan]) + ([Sales], [Feb]) + ([Sales], [Mar]) + ([Sales], [Apr]))
```

A tie situation is possible because calculated members may have the same SOLVE_ORDER value. The tie is broken based on the position of the dimensions to which the calculated members are attached:

- For aggregate storage outlines, the calculated member belonging to the dimension that comes later in the outline is the one that wins in this case.
- For block storage database outlines (and for pre-Release 7.1.2 aggregate storage outlines), the solve order property applies to calculated members defined in an MDX query. The calculated member belonging to the dimension that comes earlier in the outline is the one that wins in this case, and its expression is used to evaluate the cell.

Calculated Members

For examples of queries using calculated members, see examples for the following functions:

[Abs](#)

[Avg](#)

[BottomPercent](#)

[Case](#)

[ClosingPeriod](#)

[Count](#)

[Exp](#)

[FirstSibling](#)

[IIF](#)

[Int](#)

[Lag](#)

[LastPeriods](#)

[Lead](#)

[Ln](#)

[Max](#)

[Min](#)

[Mod](#)

[NextMember](#)

[NonEmptyCount](#)

[Ordinal](#)

[PrevMember](#)

[Remainder](#)

[Sum](#)

[ToDate](#)

Named Sets

For examples of queries using named sets, see examples for the following functions:

[BottomPercent](#)

[CurrentTuple](#)

[Filter \(example 3\)](#)

[Generate](#)

[Parent \(example 2\)](#)

Perspective

For examples of varying attribute queries using perspective, see examples for the following functions:

[AttributeEx](#)

[WithAttrEx](#)

MDX Dimension Specification

A dimension is a top-level member in the hierarchy (a member with no parent). Represent a dimension using the following rules:

Syntax

```
<dimension> ::= =  
    <dimension-name-specification>  
    | <member>.DIMENSION  
    | <layer>.DIMENSION  
    | DIMENSION ( <member> | <layer> )
```

Syntax	Description
<dimension-name-specification>	A dimension name. See Description, item 1.
<member>.DIMENSION	Dimension function with a member specification as input.
<layer>.DIMENSION	Dimension function with a layer specification as input.
DIMENSION (<member> <layer>)	Alternate syntax. Dimension (<member>) has the same effect as <member>.Dimension. Dimension (<layer>) has the same effect as <layer>.Dimension.

Description

A dimension can be represented in the following ways:

1. Using the dimension name (the name of the top member of a dimension.) For example, `[Market]`.
2. Using the `Dimension` function with a member of a dimension as input. For example, `[New York].Dimension` or `Dimension ([New York])`.
3. Using the `Dimension` function with a layer specification as input. For example, `Dimension ([Market].Generations(2).Members)` or `{ ([Market].Generations(2).Members) }.Dimension`.

MDX Layer Specification

A layer is a shared depth in the outline hierarchy. Therefore, the concept of *layer* includes generations and levels. Represent a layer using the following rules:

Syntax

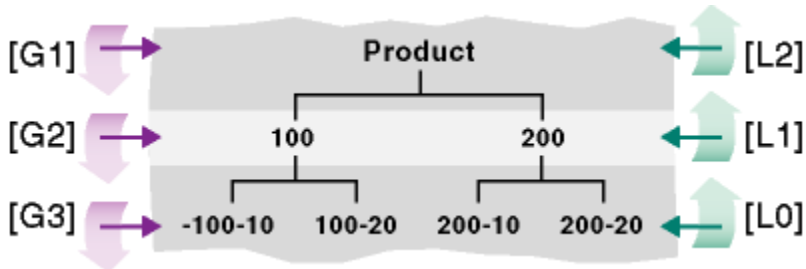
```
<layer> ::=
    <layer-name-specification>
    | Levels ( <dim_hier>, <index> )
      | <dim_hier>.Levels ( <index> )
    | Generations ( <dim_hier>, <index> )
      | <dim_hier>.Generations ( <index> )
    | <member>.Generation
    | <member>.Level
```

Syntax	Description
<layer-name-specification>	A layer name can be specified in the following ways: <ol style="list-style-type: none"> 1. By specifying the generation or level names; for example, <code>States</code> or <code>Regions</code>. The generation or level name can be within braces; for example, <code>[Regions]</code>. Using braces is recommended. 2. By specifying the dimension name along with the generation or level name; for example, <code>Market</code>, <code>Regions</code> and <code>[Market].[States]</code> This naming convention is recommended.
<dimension>.Levels (<index>)	<code>Levels</code> function with the dimension specification and a level number as input. For example, <code>[Year].Levels(0)</code> .
Levels (<dimension>, <index>)	Alternate syntax for <code>Levels</code> function with the dimension specification and a level number as input. For example, <code>Levels ([Year], 0)</code> .
<dimension>.Generations (<index>)	<code>Generations</code> function with the dimension specification and a generation number as input. For example, <code>[Year].Generations (3)</code> .
Generations (<dimension>, <index>)	Alternate syntax for <code>Generations</code> function with the dimension specification and a generation number as input. For example, <code>Generations ([Year], 3)</code> .
<member>.Generation	<code>Generation</code> function with a member specification as input. For example, <code>[Year].Generation</code> . Returns the generation of the specified member.
<member>.Level	<code>Level</code> function with a member specification as input. For example, <code>[Year].Level</code> . Returns the level of the specified member.

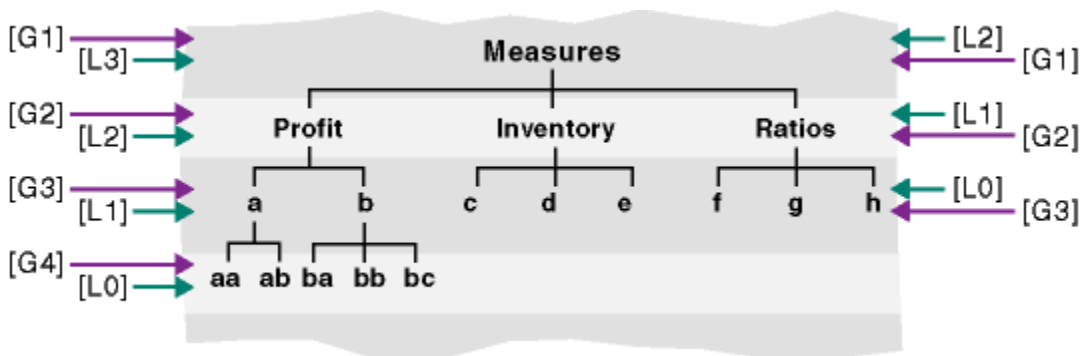
Description

Generation numbers begin counting with 1 at the dimension name; higher generation numbers are those that are closest to leaf members in a hierarchy.

Level numbers begin with 0 at the deepest part of the hierarchy; the highest level number is a dimension name.



Note: In an asymmetric (or *ragged*) hierarchy, same level numbers does *not* mean that the members are at the same depth in the outline. For example, in the following diagram, member *aa* and member *f* are both level 0 members, and yet they are not at the same depth:



MDX Member Specification

A member is a named hierarchical element in a database outline. Represent a member using the following rules:

Syntax

```
<member> ::=
    <member-name-specification>
    | <member_value_expression>
```

Member Name Specification

A member name can be specified in the following ways:

1. By specifying the actual name or the alias; for example, *Cola*, *Actual*, *COGS*, and *[100]*.

If the member name starts with number or contains spaces, it should be within braces; for example, [100]. Braces are recommended for all member names, for clarity and code readability.

If the member name starts with an ampersand (&), it should be within quotation marks; for example, ["&xyz"]. This is because the leading ampersand is reserved for substitution variables. You can also specify it as StrToMbr("&100").

For attribute members, the long name (qualified to uniquely identify the member) should be used; for example, [Ounces_12] instead of [12].

2. By specifying dimension name or any one of the ancestor member names as a prefix to the member name; for example, [Product].[100-10] and [Diet].[100-10]. This is a recommended practice for all member names, as it eliminates ambiguity and enables you to refer accurately to shared members.

Note: Use only one ancestor in the qualification. Essbase returns an error if multiple ancestors are included. For example, [Market].[New York] is a valid name for New York, and so is [East].[New York]. However, [Market].[East].[New York] returns an error.

3. By specifying the name of a calculated member defined in the WITH section.
4. For outlines that have duplicate member names enabled, see also [“MDX Syntax for Specifying Duplicate Member Names and Aliases” on page 947](#).

Member Value Expression

A member value expression is output from any function that returns a member. As an alternative to referencing the member by name or alias, you can use a function that returns a member in place of <member>. For a list of functions that return a member, see [“MDX Functions” on page 989](#).

MDX Hierarchy Specification

A hierarchy is a root member of an alternate hierarchy, which is always at generation 2 of a dimension. Member value expressions are not allowed as hierarchy arguments.

Alternate hierarchies are applicable to aggregate storage databases only.

The dimension of the hierarchy argument passed to a function must match the dimension of the other arguments passed to the function. If they do not match, an error is returned, and the query is aborted.

MDX Tuple Specification

This section shows rules for tuple specifications.

A **tuple** is a collection of member(s) with the restriction that no two members can be from the same dimension. For example, (Actual, Sales) is a tuple. (Actual, Budget) is not a tuple, as both members are from the same dimension.

Syntax

```
<tuple> ::=  
    <member>  
    | ( <member> [, <member> ].. )  
    | <tuple_value_expression>
```

Syntax	Description
<member>	A member name. If a member name contains spaces or special characters, enclose it in brackets []. It is good practice to use brackets for member names, even if they do not contain special characters. Example: [West]
(<member> [, <member>]..)	One or more member names, separated by commas. The members must be from different dimensions. The list of members must be enclosed in parentheses (). Example: ([West], [Feb])
<tuple_value_expression>	An instance of a function that extracts a tuple from a set . There are two such functions available: <ul style="list-style-type: none">● CurrentTuple● Item

Description

A tuple represents a single data cell if all dimensions are represented. For example, this tuple from Sample Basic is a single data value:

```
( [Qtr1], [Sales], [Cola], [Florida], [Actual] )
```

MDX Create Set / Delete Set

This section shows how to create and delete a named set that persists for the duration of a login session.

A named set is a re-usable member selection that can help streamline the writing and execution of MDX queries.

Syntax

The syntax to create or delete session-persistent named sets is shown below:

```
CREATE SET set_name AS ' set ' [WHERE [< slicer_specification >]]  
|DELETE set_name
```

Examples

Example 1

The following statement creates a named set called "My Favorite Customers," which is a selection of the top three customers for sales in 2001:

```
CREATE SET [My Favorite Customers] AS '{ TopCount ([Customer].Individual).Members, 3,  
([Measures].[Sales], [Time].[2001]) }'
```

The following query, issued in the same login session as the CREATE statement, references the stored named set "My Favorite Customers":

```

SELECT
{ [Time].[2000], [Time].[2001]} ON COLUMNS
{ [My Favorite Customers] } ON ROWS
FROM Sample.Basic
WHERE ( [Measures].[Profits] )

```

Example 2

To provide a context, a slicer clause may be added to the set creation statement, as shown in bold:

```

CREATE SET [My Favorite Customers] AS
' { TopCount ([Customer].Individual).Members, 3, ([Measures].[Sales], [Time].[2001])) }
  WHERE (East, Toys) '

```

Notes

- Only 16 session-based named sets may be stored simultaneously.
- Named set definitions may not contain references to other named sets.

MDX Operators

This section describes operators that can be used in MDX queries as part of numeric value expressions or search conditions.

Mathematical Operators

Operator	Definition
+	Adds. Also can be used as a unary operator.
-	Subtracts. Also can be used as a unary operator; for example, -5, -(Profit).
*	Multiplies.
/	Divides.
%	Evaluates percentage. For example, Member1%Member2 evaluates Member1 as a percentage of Member2. Note: Aggregate storage outline formulas cannot contain the % operator. In outline formulas, replace % with expression: (value1/value2)*100)

Conditional and Logical Operators

Conditional operators take two operands and check for relationships between them, returning TRUE or FALSE.

Operator	Definition
>	Data value is greater than.
<	Data value is less than.
=	Data value is equal to.

Operator	Definition
<>	Data value is not equal to.
>=	Data value is greater than or equal to.
<=	Data value is less than or equal to.
IN	<p>The syntax for the IN operator is as follows:</p> <pre><property> IN <member> <character_string_literal></pre> <p>The first argument, <property> should be an attribute property; for example, <code>Population</code> in the following example.</p> <p>The second argument, <member> or <character_string_literal>, should be an attribute member that is neither a level-0 member nor a generation-1 member; for example, <code>Medium</code> in the following example.</p> <p>Example</p> <p>The following filter evaluates the Population property (attribute) of the current member of Market dimension:</p> <pre>Filter ([Market].Members, Market.CurrentMember.Population IN Medium)</pre> <p>If the population attribute of the current member is Medium, the expression returns TRUE.</p>
IS	The IS operator syntax is as follows: <i>member1 IS member2</i> . The IS operator is equivalent to the IS function. For details and examples, see the IS function .

Boolean Operators

Boolean operators can be used in the following functions to perform conditional tests: Filter, Case, IIF, Generate. Boolean operators operate on boolean operands (TRUE/FALSE values).

See also “[MDX Functions that Return a Boolean](#)” on page 995.

Operator	Definition
AND	Logical AND linking operator for multiple value tests. Result is TRUE if both conditions are TRUE. Otherwise the result is FALSE. For an example using AND, see IsValid .
OR	Logical OR linking operator for multiple value tests. Result is TRUE if either condition is TRUE. Otherwise the result is FALSE.
NOT	Logical NOT operator. Result is TRUE if condition is FALSE. Result is FALSE if condition is TRUE. For an example using NOT, see IsEmpty .
XOR	Logical XOR linking operator for multiple value tests. Result is TRUE if <i>only</i> one condition is TRUE. Otherwise the result is FALSE.

About MDX Properties

Properties describe certain characteristics of data and metadata. MDX enables users to write queries that use properties to retrieve and analyze data. Properties can be intrinsic or custom.

“[MDX Intrinsic Properties](#)” on page 965

“[MDX Custom Properties](#)” on page 965

“[MDX Optimization Properties](#)” on page 966

[“Querying for Member Properties in MDX” on page 968](#)

[“The Value Type of MDX Properties” on page 969](#)

[“MDX NULL Property Values” on page 969](#)

MDX Intrinsic Properties

Intrinsic properties are defined for members in all dimensions. In Essbase, the intrinsic MDX member properties defined for all members in an Essbase database outline are MEMBER_NAME, MEMBER_ALIAS, LEVEL_NUMBER, GEN_NUMBER, IS_EXPENSE, COMMENTS, RELATIONAL_DESCENDANTS, and MEMBER_UNIQUE_NAME.

The MEMBER_NAME intrinsic property returns a member name string for each member.

The MEMBER_ALIAS intrinsic property returns a member alias string for each member.

The LEVEL_NUMBER intrinsic property returns the level number of each member.

The GEN_NUMBER intrinsic property returns the generation number of each member.

The IS_EXPENSE intrinsic property returns TRUE if a member has the Expense account type, and FALSE otherwise. Example:

```
SELECT
  [Measures].Members
  DIMENSION PROPERTIES [Measures].[IS_EXPENSE] on columns
from Sample.Basic;
```

The COMMENTS intrinsic property returns a comment string for each member where applicable. Example:

```
SELECT
  [Market].Members
  DIMENSION PROPERTIES [Market].[COMMENTS] on columns
from Sample.Basic;
```

The RELATIONAL_DESCENDANTS intrinsic property returns TRUE if a member has descendants in a relational database, and FALSE otherwise. This property is only applicable for Hybrid Analysis. Example:

```
SELECT
  [Market].Members
  DIMENSION PROPERTIES [Market].[RELATIONAL_DESCENDANTS] on columns
from Sample.Basic;
```

The MEMBER_UNIQUE_NAME intrinsic property is a member-name property. It returns NULL for unique members, and a system-generated key for duplicate members.

MDX Custom Properties

MDX in Essbase supports three types of custom properties: attribute properties, UDA properties, and alias-table-name properties. Attribute properties are defined by the attribute dimensions in an outline. In the Sample Basic database, the [Pkg Type] attribute dimension describes the

packaging characteristics of members in the Product dimension. This information can be queried in MDX using the property name [Pkg Type].

Attribute properties are defined only for specific dimensions and only for a specific level in each dimension. For example, in the Sample Basic outline, [Ounces] is an attribute property defined only for members in the Product dimension, and this property has valid values only for the level-0 members of the Product dimension. The [Ounces] property does not exist for other dimensions, such as Market. The [Ounces] property for a non level-0 member in the Product dimension is a NULL value. The attribute properties in an outline are identified by the names of attribute dimensions in that outline.

The custom properties also include UDAs. For example, [Major Market] is a UDA property defined on Market dimension members. It returns a TRUE value if [Major Market] UDA is defined for a member, and FALSE otherwise.

Custom alias-table-name properties enable you to query for alias table names used by each member returned in the output.

MDX Optimization Properties

Optimization properties can improve the performance of formulas and calculated members, as well as the performance of queries that rely on them.

Optimization properties are applicable to outline members with formulas and calculated members only. Stored members are not associated with these properties.

The NONEMPTYMEMBER and NONEMPTYTUPLE properties enable MDX in Essbase to query on large sets of members or tuples while skipping formula execution on non-contributing values that contain only #MISSING data.

Because large sets tend to be very sparse, only a few members contribute to the input member (have non #MISSING values) and are returned. As a result, the use of NONEMPTYMEMBER and NONEMPTYTUPLE in calculated members and formulas conserves memory resources, allowing for better scalability, especially in concurrent user environments.

NONEMPTYMEMBER

```
NONEMPTYMEMBER nonempty_member_list
```

where *nonempty_member_list* is one or more comma-separated member names or calculated member names from the same dimension as the formula or calculated member.

Use a single NONEMPTYMEMBER property clause at the beginning of a calculated member or formula expression to indicate to Essbase that the value of the formula or calculated member is empty when any of the members specified in *nonempty_member_list* are empty.

NONEMPTYTUPLE

```
NONEMPTYTUPLE "(" nonempty_member_list ") "
```

where *nonempty_member_list* is one or more comma-separated member names or calculated member names, each from different dimensions.

Use a single NONEMPTYTUPLE property clause at the beginning of a calculated member or formula expression to indicate to Essbase that the value of the formula or calculated member is empty when the cell value at the tuple given in *nonempty_member_list* is empty.

Example

The following query calculates a member [3 Month Units] that represents the sum of Units (items per package) for the current month and the previous two months, where Units data is not missing.

The calculated member [3 Month Units] calculates Units shipped for last three months. If the units shipped for [MTD] (units shipped in a year) is empty, it follows that Units data is empty for all months in the Year; therefore, the sum of Units shipped for last three months is also empty. Because the row axis in the query is very large and sparse, the NONEMPTYTUPLE property would significantly increase the performance of the query in this case.

```
WITH MEMBER [Measures].[3 Month Units] AS
,
  NONEMPTYTUPLE ([Units], [MTD])
  Sum(
    {
      ClosingPeriod(Time.Generations(5), Time.CurrentMember),
      Time.CurrentMember.Lag(1),
      Time.CurrentMember.Lag(2)
    },
    Units
  )
,
SELECT
  {Units, [3 Month Units]} ON COLUMNS,
  NON EMPTY
  CrossJoin(
    Stores.Levels(0).Members,
    [Store Manager].Children
  )
ON ROWS
FROM Asosamp.Sample
WHERE (Mar);
```

This query returns the following grid (results truncated):

(axis)	Items Per Package	3 Month Units
(017589, Carrie)	610	1808
(020408, Debra)	584	1778
(020486, Kalluri)	551	1670
(047108, Kimberley)	593	1723
(051273, Madhukar)	541	1642
(056098, Melisse)	607	1750
...

Querying for Member Properties in MDX

Properties can be used inside an MDX query in two ways. In the first approach, you can list the dimension and property combinations for each axis set. When a query is executed, the specified property is evaluated for all members from the specified dimension and included in the result set.

For example, on the column axis, the following query will return the GEN_NUMBER information for every Market dimension member. On the row axis, the query returns MEMBER_ALIAS information for every Product dimension member.

```
SELECT
  [Market].Members
  DIMENSION PROPERTIES [Market].[GEN_NUMBER] on columns,
  Filter ([Product].Members, Sales > 5000)
  DIMENSION PROPERTIES [Product].[MEMBER_ALIAS] on rows
from Sample.Basic
```

When querying for member properties using the DIMENSION PROPERTIES section of an axis, a property can be identified by the dimension name and the name of the property, or just by using the property name itself. When a property name is used by itself, that property information is returned for all members from all dimensions on that axis, for which that property applies.

Note: When a property name is used by itself within the DIMENSION PROPERTIES section, do not use brackets [] around the property name.

In the following query, the MEMBER_ALIAS property is evaluated on the row axis for both Year and Product dimensions.

```
SELECT
  [Market].Members
  DIMENSION PROPERTIES [Market].[GEN_NUMBER] on columns,
  CrossJoin([Product].Children, Year.Children)
  DIMENSION PROPERTIES MEMBER_ALIAS on rows
from Sample.Basic
```

In a second approach, properties can be used inside value expressions in an MDX query. For example you can filter a set based on a value expression that uses properties of members in input set.

The following query returns all caffeinated products that are packaged in cans.

```
Select
Filter([Product].levels(0).members,
  [Product].CurrentMember.Caffeinated and
  [Product].CurrentMember.[Pkg Type] = "Can")
Dimension Properties
  [Caffeinated], [Pkg Type] on columns
```

The following query uses the UDA [Major Market] to calculate the value [BudgetedExpenses] based on whether the current member of the Market dimension is a major market or not.


```

With
    MEMBER [Measures].[BudgetedExpenses] AS
        'IIF([Market].CurrentMember.[Major Market],
            [Marketing] * 1.2, [Marketing])'
Select
    {[Measures].[BudgetedExpenses]} on columns,
    Market.Members on rows
Where
    ([Budget])

```

The following queries use alias table names.

```

SELECT
    [Product].Members
    DIMENSION PROPERTIES [Default] on columns
from Sample.Basic;

```

```

SELECT
    [Product].Members
    DIMENSION PROPERTIES [Long Names] on columns
from Sample.Basic;

```

The Value Type of MDX Properties

The value of an MDX property in Essbase can be a numeric, Boolean, or string type. MEMBER_NAME and MEMBER_ALIAS properties return string values. LEVEL_NUMBER and GEN_NUMBER properties return numeric values.

The attribute properties return numeric, Boolean, or string values based on the attribute dimension type. For example, in Sample Basic, the [Ounces] attribute property is a numeric property. The [Pkg Type] attribute property is a string property. The [Caffeinated] attribute property is a Boolean property.

Essbase allows attribute dimensions with date types. The date type properties are treated as numeric properties in MDX. When comparing these property values with dates, you need to use the TODATE function to convert date strings to numeric before comparison.

The following query returns all Product dimension members that have been introduced on date 03/25/1996. Since the property [Intro Date] is a date type, the TODATE function must be used to convert the date string "03-25-1996" to a number before comparing it.

```

Select
    Filter ([Product].Members,
        [Product].CurrentMember.[Intro Date] =
            TODATE("mm-dd-yyyy", "03-25-1996")) on columns

```

When a property is used in a value expression, you must use it appropriately based on its value type: string, numeric, or Boolean.

MDX NULL Property Values

Not all members may have valid values for a given property name. For example, the MEMBER_ALIAS property returns an alternate name for a given member as defined in the

outline; however, not all members may have aliases defined. In these cases A NULL value would be returned for those members that do not have aliases.

In the following query:

```
SELECT
  [Year].Members
  DIMENSION PROPERTIES MEMBER_ALIAS on columns
```

none of the members in the Year dimension have aliases defined for them. Therefore, the query returns NULL values for the MEMBER_ALIAS property for members in the Year dimension.

The attribute properties are defined for members of a specific dimension and a specific level in that dimension. In the Sample Basic database, the [Ounces] property is defined only for level-0 members of the Product dimension.

Therefore, if you query for the [Ounces] property of a member from the Market dimension, as shown in the following query, you will get a syntax error:

```
SELECT
  Filter([Market].members,
    [Market].CurrentMember.[Ounces] = 32) on columns
```

Additionally, if you query for the [Ounces] property of a non level-0 member of the dimension, you will get a NULL value.

When using property values in value expressions, you can use the function IsValid() to check for NULL values. The following query returns all Product dimension members with [Ounces] property value of 12, after eliminating members with NULL values.

```
Select
  Filter([Product].Members,
    IsValid([Product].CurrentMember.[Ounces]) and
    [Product].CurrentMember.[Ounces] = 12) on columns
```

MDX Comments

This section describes how to add comments to MDX queries.

Syntax

MDX supports two types of syntax for comments:

1. MDX supports the "C++ style" comments that are also supported by the Essbase Server calculator framework. This type of comment can cover multiple lines. Everything in between is ignored by the MDX parser.

Example:

```
/*
commented text is
ignored by parser
*/
```

- MDX supports inline comments beginning with two hyphens. Beginning with two hyphens, the rest of the line is ignored by the MDX parser. A new line ends the span of the comment.

Example:

```
-- short comment can go on till line break
```

Example

The following example uses both styles of comments:

```
/* Query the profit figures in each
   market for the "100" products
*/
SELECT
  {[Market].levels(1).members}  --L1 members of Market
ON COLUMNS,
  --Cross of the "100" products and their profit figures:
  CrossJoin ([100].children, [Profit].children)
ON ROWS
FROM Sample.Basic
```

MDX Query Limits

Overview

The following concepts are applicable to understanding MDX query limits.

Concept	Description
NON EMPTY processing	Refers to how Essbase processes MDX queries and sets when the NON EMPTY keywords are used in an axis specification . The NON EMPTY specification optimizes processing by suppressing slices that would contain entirely #MISSING values.
Cluster elements / symmetric sets	<p>Although an MDX set is a collection of tuples, internally, Essbase represents sets using clusters and tuples. A cluster is a type of set derived using the CrossJoin function, where the arguments to CrossJoin are sets from one dimension only.</p> <p>A cluster can also be thought of as a symmetric set. The following set is a symmetric set and can be stored as one cluster.</p> <pre>CROSSJOIN(Products.LEVELS(0).MEMBERS, [Market].LEVELS(0).MEMBERS)</pre> <p>A tuple is a collection of members from different dimensions. The following set has one tuple.</p> <pre>{([Product].Product_1, [Market].Market_1)}</pre> <p>The following set is a union of the above two sets. It is stored internally as a cluster and a tuple.</p> <pre>UNION(CROSSJOIN(Products.LEVELS(0).MEMBERS, [Market].LEVELS(0).MEMBERS) , {([Product].Product_1, [Market].Market_1)})</pre>
Compact set	A set is stored in compact form if it can be internally represented as a cluster or symmetric set.

Concept	Description
Flattened set	<p>A set that must be internally expanded into tuples is a flattened set. Flattened sets consume more memory to be processed. Certain MDX functions, such as Order, need to flatten sets in order to process them correctly. Therefore, certain functions, as listed in the next section, have different set size or query limits.</p> <p>The following set is an example of a flattened set.</p> <pre>{ (Colas, East) (Colas, West) (Colas, South) (Colas, Central) (Root Beer, East) (Root Beer, West) (Root Beer, South) (Root Beer, Central) (Cream Soda, East) (Cream Soda, West) (Cream Soda, South) (Cream Soda, Central) (Fruit Soda, East) (Fruit Soda, West) (Fruit Soda, South) (Fruit Soda, Central) }</pre>
Asymmetric set	<p>The following set is stored internally as a collection of a tuple element and a cluster element. The two elements cannot be combined into a single element. Such sets are called asymmetric sets.</p> <pre>UNION({ (Colas, East) } CROSSJOIN([Product].CHILDREN, [Market].CHILDREN))</pre>

MDX Query Limits

The following size limitations apply to MDX queries, sets, and certain functions.

Note: The following exception applies to the general query limits: If the database being queried is the target database of a partition, the maximum size of a cube region you can query using MDX is 2^{32} potential cells.

Limitations	Units
Number of cells in a query region defined by all axis sets in an MDX query with NON EMPTY clause	2^{64}
Number of cells that can be returned to a client after NON EMPTY processing	2^{32}
Number of cells in a query region defined by all axis sets in an MDX query with no NON EMPTY clause	2^{32}
Number of tuples in an axis set with NON EMPTY directive after NON EMPTY processing	2^{28}

Limitations	Units
Size of a set in compact form	2^{64}
Size of a set in flattened form	2^{32}
Number of elements in a set	2^{32}
Number of members (from all dimensions) in a cluster element	2^{32}
Number of cells in a query after applying non empty cell processing	2^{32}
Size of a set that can be processed by the following functions: <ul style="list-style-type: none"> ● Distinct ● Except ● Filter ● Intersect ● Ntile ● Order ● Percentile ● Rank ● TopPercent ● BottomPercent ● TopSum ● BottomSum ● Hierarchize ● Union (with removal of duplicates) ● NonEmptySubset (output set size) ● TopCount (output set size) ● BottomCount (output set size) 	2^{28}
EssMdx API	<ul style="list-style-type: none"> ● Maximum number of tuples/clusters on an axis—2^{29-1} ● Maximum number of cells (when cell status is not requested)—2^{27-1} ● Maximum number of cells (when cell status is not requested)—approximately 2^{26-1}
MDX queries run through MaxL	<ul style="list-style-type: none"> ● Maximum number of columns—2^{29-1} ● Maximum number of rows—2^{29-1}

Aggregate Storage and MDX Outline Formulas

To write formulas for block storage outlines, Essbase provides a set of calculation functions and operators known as the Calculator, or Calc, language. The Calculator language cannot be used

to write member formulas for aggregate storage databases. Formulas in aggregate storage outlines use the MDX language.

The following sections provide information for rewriting Calculator formulas in MDX for outlines that have been migrated from block storage to aggregate storage. Before attempting to rewrite formulas you should be familiar with the basic workings of aggregate storage outlines in Essbase. See the *Oracle Essbase Database Administrator's Guide*, which discusses all aspects of aggregate storage.

Translating Calculator Functions to MDX Functions

When translating Calculator formulas to MDX, keep in mind the following differences between block storage outlines and aggregate storage outlines:

- The storage characteristics of a member and hence all its associated cells are defined in a block storage outline through Dynamic Calc (and Dynamic Calc and Store) attributes, and stored attributes. Such attributes do not exist in an aggregate storage outline. Upper level members along an explicitly tagged accounts dimension and members with formulas attached to them are always calculated dynamically in such a database.
- In block storage outlines, calculation order is dependent on the order in which members appear in the outline whereas formulas are executed in order of their dependencies in aggregate storage outlines. In addition, calculation order in the event of ambiguity in the evaluation of a cell, and two-pass calculation tags are not required in an aggregate storage outline.
- The layout of block storage outlines and the separation of dimensions into dense and sparse has an effect on the semantics of certain calculations, giving rise to concepts such as top-down calculation mode, cell and block calculation mode, and create-blocks on equations. The simplicity of the aggregate storage outlines, which do not separate dimensions into dense and sparse, do not require such concepts.

General Guidelines for Translating Calculator Formulas to MDX

This section provides some general guidelines for translating Calculator formulas to MDX.

Be certain that the application has been redesigned to use an aggregate storage outline. In this regard, make certain that formulas do not reference any block-storage specific outline constructs, such as variance functions that rely on expense tagging, or functions that operate on shared members (for example, @RDESCENDANTS). Such constructs are not valid in aggregate storage outlines.

Rewrite each function in the formulas attached to an explicitly tagged accounts dimension for which a direct counterpart in MDX exists. [Table 1](#) provides specific information and examples. Then identify functions for which an indirect rewrite is required. [Table 1](#) also provides information and examples for these functions.

Understand the calculation order semantics for the formulas in the block storage outline. Organize the dependent formulas in the aggregate storage outline carefully to achieve the same results as block storage.

If formulas reference custom-defined functions or macros consider rewriting them, if possible, using other MDX functions.

The following table lists all functions in the Calculator language and their analogs in MDX (and vice versa). Where a direct analog does not exist, transformation rules and examples are provided.

Table 1 Calculator to MDX Function Mapping

Calculator	MDX	Remarks/Examples
@ABS	Abs	<p>Calculator</p> <p>@ABS (Actual-Budget)</p> <p>MDX</p> <p>Abs ([Actual] - [Budget])</p>
@ALLANCESTORS	Ancestors	Shared members are not relevant to aggregate storage outlines.
@ALIAS	Not required.	In MDX, the argument to @ALIAS can be passed as-is to the outer function.
@ANCEST	Ancestor with CurrentMember as input. Use a tuple to combine the result with the optional third argument to the @ANCEST function.	<p>Calculator</p> <p>@ANCEST (Product, 2, Sales)</p> <p>MDX</p> <pre>(Sales, Ancestor(Product.CurrentMember, Product.Generations(2)))</pre>
@ANCESTORS	Ancestors	<p>Calculator</p> <p>@ANCESTORS ("New York")</p> <p>MDX</p> <p>Ancestors ([New York].parent, [Market].levels(2))</p>
@ANCESTVAL	Ancestor with CurrentMember as input. Use a tuple to combine the result with the optional third argument to the @ANCESTVAL function.	<p>Calculator</p> <p>@ANCESTVAL (Product, 2, Sales)</p> <p>MDX</p> <pre>(Sales, Ancestor(Product.CurrentMember, Product.Generations(2))).Value</pre>

Calculator	MDX	Remarks/Examples
@ATTRIBUTE	Attribute	<p>Calculator</p> <p>@ATTRIBUTE (Can)</p> <p>MDX</p> <p>Attribute ([Can])</p>
@ATTRIBUTEVAL	[BaseDim] .CurrentMember. AttributeDim	<p>See “About MDX Properties” on page 964. Calculator</p> <p>@ATTRIBUTEVAL (Caffeinated)</p> <p>MDX</p> <p>Product .CurrentMember .Caffeinated</p>
@ATTRIBUTESVAL	[BaseDim] .CurrentMember. AttributeDim	<p>See “About MDX Properties” on page 964. Calculator</p> <p>@ATTRIBUTESVAL (" Pkg Type ")</p> <p>MDX</p> <p>Product .CurrentMember . [Pkg Type]</p>
@ATTRIBUTEVAL	[BaseDim] .CurrentMember AttributeDim	<p>See “About MDX Properties” on page 964. Calculator</p> <p>@ATTRIBUTEVAL (Ounces)</p> <p>MDX</p> <p>Product .CurrentMember . Ounces</p>
@AVG	<p>If the dimensionality of all elements in the input set to @AVG is the same, use Avg. Translate SKIPNONE to INCLUDEEMPTY.</p> <p>If the dimensionality of all elements in the input set to @AVG is not the same, then perform average by explicitly adding the tuples and dividing by the set cardinality (the number of tuples in the set).</p>	<p>Note that the MDX Avg function skips missing cell values by default.</p> <p>Calculator</p> <p>@AVG (SKIPMISSING, @CHILDREN (East))</p> <p>MDX</p> <p>Avg ([East] .Children)</p> <p>If SKIPMISSING is replaced by SKIPNONE, the translation changes to:</p> <p>Avg ([East] .Children, Sales, INCLUDEEMPTY)</p> <p>For SKIPZERO, the translation is:</p> <p>Avg ([East] .Children, IIF (Market .CurrentMember .Value=0, Missing, IIF (Market .CurrentMember = Missing, 0, Market .CurrentMember .Value)))</p> <p>For SKIPBOTH, the translation is:</p> <p>Avg ([East] .Children, IIF (Market .CurrentMember=0, Missing, Market .CurrentMember .Value))</p>

Calculator	MDX	Remarks/Examples
@AVGRANGE	CrossJoin (first argument, set created out of second argument). The rest is similar to @AVG when the dimensionality of all elements of the input set is identical.	<p>Calculator</p> <pre>@AVGRANGE(SKIPMISSING, Sales, @CHILDREN(West))</pre> <p>MDX</p> <pre>Avg(CrossJoin({Sales}, {[West].Children}))</pre> <p>If SKIPMISSING is replaced by SKIPNONE, the translation becomes:</p> <pre>Avg({[West].Children}, Sales, INCLUDEEMPTY)</pre> <p>If SKIPZERO is used, then the translation is:</p> <pre>Avg([West].Children, IIF(Sales = 0, Missing, IIF(Sales = Missing, 0, Sales)))</pre>
@CHILDREN	Children	<p>Calculator</p> <pre>@CHILDREN(Market)</pre> <p>MDX</p> <pre>Children(Market)</pre> <p>or</p> <pre>Market.Children</pre>
@CONCATENATE	Concat	<p>Calculator</p> <pre>@MEMBER(@CONCATENATE("Qtr1", "1"));</pre> <p>MDX</p> <pre>Concat("01", "01")</pre>
@CORRELATION	Not supported in MDX.	.

Calculator	MDX	Remarks/Examples
@COUNT	<p>Use Count if SKIPNONE.</p> <p>Use NonEmptyCount if SKIPMISSING.</p> <p>For SKIPZERO, see the example in the next column.</p> <p>For SKIPBOTH, use Count (Filter(set, value <> 0 && value <> MISSING))</p>	<p>Calculator</p> <pre>@COUNT (SKIPMISSING, @RANGE (Sales, Children (Product)))</pre> <p>MDX</p> <pre>NonEmptyCount (CrossJoin ({Sales}, {Product.Children}))</pre> <p>Note that Count always counts including the empty cells, whereas NonEmptyCount does not.</p> <p>For SKIPNONE, the translation is:</p> <pre>Count (Product.Children)</pre> <p>For SKIPZERO, the translation is:</p> <pre>NonEmptyCount (Product.Children, IIF (Sales=0, Missing, IIF (Sales = Missing, 0, sales)))</pre>
@CURGEN	Generation (CurrentMember (dimension))	<p>Calculator</p> <pre>@CURGEN (Year)</pre> <p>MDX</p> <pre>Year.CurrentMember.Generation</pre>
@CURLEV	Level (CurrentMember (dimension))	<p>Calculator</p> <pre>@CURLEV (Year)</pre> <p>MDX</p> <pre>Year.CurrentMember.Level</pre>
@CURRMBR	CurrentMember	<p>Calculator</p> <pre>@CURRMBR (Product)</pre> <p>MDX</p> <pre>[Product].CurrentMember</pre>
@CURRMBRRANGE	RelMemberRange	<p>Calculator</p> <pre>@CURRMBRRANGE (Year, LEV, 0, -1, 1)</pre> <p>MDX</p> <pre>RelMemberRange (Year.CurrentMember, 1, 1, LEVEL)</pre>
@DESCENDANTS	Descendants (member)	See MDX Descendants documentation for examples.

Calculator	MDX	Remarks/Examples
@EXP	Exp	Calculator @EXP("Variance %"/100); MDX Exp([Scenario].[Variance %]/100)
@FACTORIAL	Factorial	Calculator @FACTORIAL(5) MDX Factorial(5)
@GEN, @LEV	Generation, Level	.
@GENMBRS, @LEVMBRS	layer.Members	.
@IALANCESTORS	Ancestors	Shared members are not relevant to aggregate storage outlines.
@IANCESTORS	Ancestors	Shared members are not relevant to aggregate storage outlines.
@ICHILDREN	Union(<i>member, member.Children</i>)	Calculator @ICHILDREN(Market) MDX Union({Market}, {Market.children})
@IDESCENDANTS	Descendants(<i>member</i>)	Calculator @IDESCENDANTS(Market) MDX Descendants(Market)
@ILSIBLINGS	MemberRange (<i>member.FirstSibling,member</i>)	Calculator @ILSIBLINGS(Florida) MDX MemberRange(Florida.FirstSibling, Florida.Lag(1))
@INT	Int	Calculator @INT(104.504) MDX Int(104.504)
@ISACCTYPE	IsAccType	See MDX IsAccType documentation for examples.

Calculator	MDX	Remarks/Examples
@ISANCEST	IsAncestor	<p>Calculator</p> <pre>@ISANCEST(California)</pre> <p>MDX</p> <pre>IsAncestor(Market.CurrentMember, California)</pre>
@ISCHILD	IsChild	See MDX IsChild documentation for examples.
@ISDESC	See examples.	<p>Calculator</p> <pre>@ISDESC(Market)</pre> <p>MDX</p> <pre>IsAncestor([Market], [Market].Dimension.CurrentMember)</pre> <p>or</p> <pre>Count(Intersect({Member.Descendants}, {Member.dimension.CurrentMember})) = 1</pre>
@ISGEN	IsGeneration	<p>Calculator</p> <pre>@ISGEN(Market, 2)</pre> <p>MDX</p> <pre>IsGeneration(Market.CurrentMember, 2)</pre>
@ISIANCEST	IIF(Is(member, ancestormember) OR IsAncestor(member, ancestormember), <true-part>, <false-part>)	<p>Calculator</p> <pre>@ISIANCEST(California)</pre> <p>MDX</p> <pre>IIF(IS(Market.CurrentMember, California) OR IsAncestor(Market.CurrentMember, California), <true-part>, <false-part>)</pre>
@ISIBLINGS	Siblings(member)	Returns a set that includes the specified member and its siblings.
@ISICHILD	IIF(Is(member, childmember) OR IsChild(member, childmember), <true-part>, <false-part>)	<p>Calculator</p> <pre>@ISICHILD(South)</pre> <p>MDX</p> <pre>IIF(Is(Market.CurrentMember, South) OR IsChild(Market.CurrentMember, South), <true-part>, <false-part>)</pre>

Calculator	MDX	Remarks/Examples
@ISIDESC	See examples.	<p>Calculator</p> <pre>@ISIDESC(South)</pre> <p>MDX</p> <pre>(Count(Intersect({[South].Descendants}, {South})) = 1 OR Is(CurrentMember, [South]))</pre>
@ISIPARENT	IIF(Is(member, parentmember)	<p>Calculator</p> <pre>@ISIPARENT(Qtr1)</pre> <p>MDX</p> <pre>IIF(Is(Time.CurrentMember, [Qtr1]) OR IsChild([Qtr1], Time.CurrentMember), <true-part>, <false-part>)</pre>
@ISISIBLING	IsSibling(member, siblingmember)	<p>Calculator</p> <pre>@ISISIBLING(Qtr2)</pre> <p>MDX</p> <pre>IIF(IsSibling([Qtr2], Time.CurrentMember), <true-part>, <false-part>)</pre>
@ISLEV	IsLevel	.

Calculator	MDX	Remarks/Examples
@ISMBR	<code>IIF(Count(Intersect (member-set, member)) = 1, true-part, false-part)</code>	<p>Calculator allows a collection of members or cross members that do not subscribe to the rules of an MDX set to appear as the second argument. This functionality cannot be easily replicated without enumerating each element of the second set and testing for intersection.</p> <p>However, if the second argument subscribes to MDX set rules then the translation is easier, as shown. For example:</p> <p>Calculator</p> <pre>@ISMBR("New York": "New Hampshire")</pre> <p>MDX</p> <pre>IIF (Count (Intersect ({MemberRange ([New York], [New Hampshire])}, {Market.CurrentMember})) = 1, <true-part>, <false-part>)</pre>
@ISPARENT	Use <code>IsChild</code> .	<p>Calculator</p> <pre>@ISPARENT("New York")</pre> <p>MDX</p> <pre>IsChild(Market.CurrentMember, [New York])</pre>
@ISSAMEGEN, @ISSAMELEV	<code>IIF (member.Generation = CurrentMember(dimension).Generation, <true-part>, <false-part>)</code>	<p>Calculator</p> <pre>@ISSAMEGEN(West)</pre> <p>MDX</p> <pre>IIF (Ordinal (Market.CurrentMember.Generation) = Ordinal (West.Generation), <true-part>, <false-part>)</pre>
@ISSIBLING	<code>IsSibling</code>	See MDX <code>IsSibling</code> documentation for examples.
@ISUDA	<code>IsUda</code>	See MDX <code>IsUda</code> documentation for examples.
@LIST	.	If the member set does not subscribe to MDX set rules, then explicit enumeration is required. For <i>rangelist</i> use <code>CrossJoin(member, set)</code> .
@LN, @LOG, @LOG10	<code>Ln, Log, Log10</code>	.

Calculator	MDX	Remarks/Examples
@LSIBLINGS @RSIBLINGS	<pre>MemberRange(member. FirstSibling, member. Lag(1)) MemberRange(member.Lead(1), member.LastSibling)</pre>	<p>Calculator</p> <pre>@LSIBLINGS (Qtr4)</pre> <p>MDX</p> <pre>MemberRange ([Qtr4].FirstSibling, [Qtr4].Lag(1))</pre> <p>Calculator</p> <pre>@RSIBLINGS (Qtr1)</pre> <p>MDX</p> <pre>MemberRange ([Qtr1].Lead(1), [Qtr1].LastSibling)</pre>
@MATCH	.	.
@MAX	Max	<p>Use Max if argument list is a set. Otherwise, rewrite logic using Case constructs by explicit enumeration of the argument list.</p> <p>Calculator</p> <pre>@MAX (Jan:Mar)</pre> <p>MDX</p> <pre>Max (MemberRange ([Jan], [Mar]))</pre>
@MAXRANGE	Max	<p>Calculator</p> <pre>@MAXRANGE (Sales, @CHILDREN (Qtr1))</pre> <p>MDX</p> <pre>Max (CrossJoin ({Sales}, {[Qtr1].Children}))</pre> <p>OR</p> <pre>Max ([Qtr1].Children, Sales)</pre>

Calculator	MDX	Remarks/Examples
@MAXS	Max	<p>Calculator</p> <pre>@MAXS (SKIPMISSING, Sales, @CHILDREN(Qtr1))</pre> <p>MDX</p> <pre>Max (Children ([Qtr1]), Sales)</pre> <p>For SKIPZERO, the translation is:</p> <pre>Max (Children ([Qtr1]), IIF (Sales = 0, MISSING, Sales))</pre> <p>For SKIPBOTH, the translation is the same as for SKIPZERO, because Max skips missing values by default.</p>
@MAXSRANGE	Max	<p>Calculator</p> <pre>@MAXSRANGE (SKIPMISSING, Sales, @CHILDREN(Qtr1))</pre> <p>MDX</p> <pre>Max (Children ([Qtr1]), Sales)</pre> <p>For SKIPZERO, the translation is:</p> <pre>Max (Children ([Qtr1]), IIF (Sales = 0, MISSING, Sales))</pre> <p>For SKIPBOTH, the translation is the same as for SKIPZERO, because Max skips missing values by default.</p>
@MDANCESTVAL	Use Ancestor , Value , and Currentmember as shown in the example.	<p>Calculator</p> <pre>@MDANCESTVAL (2, Market, 2, Product, 2, Sales)</pre> <p>MDX</p> <p>Construct a tuple consisting of Sales from the Measures dimension, the ancestor of the current member along the Market dimension, and the ancestor of the current member along the Product dimension. Then get the value of the tuple.</p> <pre>(Sales, Ancestor (Market.CurrentMember, 2), Ancestor (Product.CurrentMember, 2)).Value</pre>

Calculator	MDX	Remarks/Examples
@MDPARENTVAL	Use Parent , Value , and CurrentMember as shown in the example.	<p>Calculator</p> <pre>@MDPARENTVAL(2, Market, Product, Sales)</pre> <p>MDX</p> <p>Construct a tuple consisting of Sales from the Measures dimension, the parent of the current member along the Market dimension, and the parent of the current member along the Product dimension. Then get the value of the tuple.</p> <pre>(Sales, Market.CurrentMember.Parent, Product.CurrentMember.Parent).Value</pre>
@MDSHIFT	See MDX equivalent for @NEXT , and repeat it for each dimension that needs to be shifted. CrossJoin the results from each dimension and get the value of the final tuple. See comments for @MDANCESTVAL .	.
@MEDIAN	Not supported in MDX.	.
@MEMBER	Not needed in MDX.	.
@MERGE	Union (set1,set2)	<p>If the lists specified as inputs to @MERGE do not subscribe to the rules of an MDX set, then the @MERGE function cannot be translated. The following example assumes that the lists do subscribe to MDX set rules.</p> <p>Calculator</p> <pre>@MERGE(@CHILDREN(East), @CHILDREN(West))</pre> <p>MDX</p> <pre>{Union([East].Children, [West].Children)}</pre>
@MIN	Min	<p>Use Min if argument list is a set. Otherwise, rewrite logic using Case constructs by explicit enumeration of the argument list.</p> <p>Calculator</p> <pre>@MIN(Jan:Mar)</pre> <p>MDX</p> <pre>Min(MemberRange([Jan], [Mar]))</pre>

Calculator	MDX	Remarks/Examples
@MINRANGE	Min	<p>Calculator</p> <pre>@MINRANGE(Sales, @CHILDREN(Qtr1))</pre> <p>MDX</p> <pre>Min(CrossJoin({Sales}, {[Qtr1].Children}))</pre> <p>OR</p> <pre>Min([Qtr1].Children, Sales)</pre>
@MINS	Min	<p>Calculator</p> <pre>@MINS(SKIPMISSING, Sales, @CHILDREN(Qtr1))</pre> <p>MDX</p> <pre>Min(Filter(Children([Qtr1]), Sales <> Missing))</pre> <p>For SKIPZERO, the translation is:</p> <pre>Min(Filter(Children([Qtr1]), Sales <> 0))</pre> <p>For SKIPBOTH, the translation is:</p> <pre>Min(Filter(Children([Qtr1]), Sales <> 0 AND Sales <> Missing))</pre>

Calculator	MDX	Remarks/Examples
@MINSRANGE	Min	<p>Calculator</p> <pre>@MINSRANGE(SKIPMISSING, Sales, @CHILDREN(Qtr1))</pre> <p>MDX</p> <pre>Min(Filter(Children([Qtr1]), Sales <> Missing))</pre> <p>For SKIPZERO, the translation is:</p> <pre>Min(Filter(Children([Qtr1]), Sales <> 0))</pre> <p>For SKIPBOTH, the translation is:</p> <pre>Min (Filter(Children([Qtr1]), Sales <> 0 AND Sales <> Missing))</pre>
@MOD	Mod	.
@MODE	Not supported in MDX.	.
@NAME	Not needed in MDX.	.
@NEXT	<p>@NEXT(member,[n, range]) returns the <i>n</i>th cell value in the range from the supplied member. The function returns a missing value if the supplied member does not exist in the range. If range is not specified, level-0 members of the Time dimension are used.</p> <p>MDX does not have an equivalent function for an arbitrary range. However, if the range is restricted to members from a specific level or generation, then using NextMember (if <i>n=1</i>) or Lead/Lag will work as shown in the sample translation. This is probably the common case.</p>	<p>Calculator</p> <pre>@Next (Cash)</pre> <p>MDX</p> <pre>(NextMember ([Year].CurrentMember, LEVEL), [Cash]).Value</pre> <p>Alternative:</p> <p>Calculator</p> <pre>@Next (Cash, 2)</pre> <p>MDX</p> <pre>CrossJoin(Year.CurrentMember.Lead(2, LEVEL), Cash).Value</pre>
@NEXTS	Not supported in MDX.	.
@PARENT	Parent	.

Calculator	MDX	Remarks/Examples
@PARENTVAL	Parent with CurrentMember as input. Use a tuple to combine the result with the optional second argument to the @PARENTVAL function.	<p>Calculator</p> <pre>@PARENTVAL(Market, Sales)</pre> <p>MDX</p> <pre>([Sales], [Market].CurrentMember.Parent).Value</pre>
@POWER	Power	.
@PRIOR	<p>@PRIOR(member,[n, range]) returns the nth cell value in the range from the supplied member. The function returns a missing value if the supplied member does not exist in the range. If range is not specified, level-0 members of the Time dimension are used.</p> <p>MDX does not have an equivalent function for an arbitrary range. However, if the range is restricted to members from a specific level or generation, then using PrevMember (if n=1) or Lead/Lag will work as shown in the sample translation. This is probably the common case.</p>	<p>Calculator</p> <pre>@Prior(Cash)</pre> <p>MDX</p> <pre>PrevMember(Year.CurrentMember, LEVEL), [Cash]).Value</pre> <p>Alternative:</p> <p>Calculator</p> <pre>@Prior(Cash, 2)</pre> <p>MDX</p> <pre>(Year.CurrentMember.Lag(2, LEVEL), [Cash]).Value</pre>
@PRIORS	Not supported in MDX.	.
@RANGE	CrossJoin(member, rangeset)	<p>Calculator automatically uses level-0 members of the Time dimension if a range is unspecified. That feature does not exist in MDX, so you must explicitly include the range.</p> <p>Calculator</p> <pre>@RANGE(Sales, @CHILDREN(East))</pre> <p>MDX</p> <pre>CrossJoin({Sales}, {[East].Children})</pre>
@RANK	Not supported in MDX. This is a vector function.	.
@REMAINDER	Remainder	.
@REMOVE	Except(set1, set2)	<p>Translation will work only if set1 and set2 are true MDX sets.</p> <p>Calculator</p> <pre>@REMOVE(@CHILDREN(East), @LIST("New York", Connecticut))</pre> <p>MDX</p> <pre>Except ({[East].Children}, {[New York], [Connecticut]})</pre>

Calculator	MDX	Remarks/Examples
@ROUND	Round	.
@SHIFT	See @PRIOR and @NEXT.	.
@SIBLINGS	Siblings	.
@STDEV, @STDEVP, @STDEV RANGE	Not supported in MDX.	.
@SUBSTRING	Not supported in MDX.	.
@SUM	Sum	Convert each element of the <i>explist</i> to a tuple so that collectively the tuples can form a set.
@SUMRANGE	Sum(<i>CrossJoin(member, Xrangelist)</i>)	<p>Calculator</p> <p>@SUMRANGE("New York", Jan:Jun)</p> <p>MDX</p> <p>Sum(CrossJoin({ [New York] }, { [Jan]:[Jun] }))</p>
@TODATE	Todate	.
@TRUNCATE	Truncate	.
@UDA	Uda	.
@VAR, @VARPER	Arg1 - Arg2	An aggregate storage outline has no expense tags. Therefore, variance functionality defaults to subtraction.
@VARIANCE, @VARIANCEP	Not supported in MDX.	.
@WITHATTR	WithAttr	.
@XRANGE	Not supported in MDX.	.
@XREF	Not supported in MDX.	.

MDX Functions

Functions can be used to generate metadata and/or value information that you need to pass to a SELECT statement. Becoming proficient with the functions reduces the need to enumerate tuples, members, numeric values, or other needed values explicitly in the set specifications of a query. More importantly, using functions allows in-depth analysis of your database.

This section contains a listing of query functions by return value. The possible return values are described in these topics:

- [“MDX Functions that Return a Member” on page 990](#)
- [“MDX Functions that Return a Set” on page 991](#)
- [“MDX Functions that Return a Tuple” on page 993](#)

- “MDX Functions that Return a Number” on page 993
- “MDX Functions that Return a Dimension” on page 995
- “MDX Functions that Return a Layer” on page 995
- “MDX Functions that Return a Boolean” on page 995
- “MDX Functions that Return a Date” on page 996
- “MDX Functions that Return a String” on page 996

MDX Functions that Return a Member

The following functions return a [member](#) or a member value expression.

Function	Result
Ancestor	Returns a member that is an ancestor of the specified member, at a specified generation or level.
ClosingPeriod	Returns the last descendant of a layer, or the last child of the Time dimension.
Cousin	Returns a child member at a matching outline level and location as a member from another parent.
CurrentMember	Returns the current member in the input dimension. <i>Current</i> is in the context of query execution mechanics. Use in combination with iterative functions such as Filter .
DateToMember	Returns the date-hierarchy member specified by the input date.
DefaultMember	Returns the default member in the input dimension.
FirstChild	Returns the first child of the input member.
FirstSibling	Returns the first child of the input member's parent.
Lag	Using the default order of members in a database outline, returns a member that is <i>n</i> steps behind the input member.
LastChild	Returns the last child of the input member.
LastSibling	Returns the last child of the input member's parent.
Lead	Using the default order of members in a database outline, returns a member that is <i>n</i> steps past the input member.
NextMember	Returns the member (in the same layer) that is one step past the input member.
OpeningPeriod	Returns the first descendant of a layer, or the first child of the Time dimension.
ParallelPeriod	Returns a member from a prior time period as the specified or default time member.
Parent	Returns a member's parent.
PrevMember	Returns the member (in the same layer) that is one step prior to the input member.
StrToMbr	Converts a string to a member name.

MDX Functions that Return a Set

The following categories of functions return a [set](#) or a set value expression.

- [Pure Set Functions](#)
- [Metadata-based Set Functions](#)
- [Data-based Set Functions](#)

Pure Set Functions

Functions in this category derive their results without getting any further information from the cube.

Function	Result
CrossJoin	Returns a cross-section of two sets from different dimensions.
Distinct	Deletes duplicate tuples from a set.
Except	Returns a subset containing the differences between two sets.
Generate	For each tuple in <i>set1</i> , return <i>set2</i> .
Head	Returns the first <i>n</i> members or tuples present in a set.
Intersect	Returns the intersection of two input sets.
Subset	Returns a subset from a set, in which the subset is a numerically specified range of tuples.
Tail	Returns the last <i>n</i> members or tuples present in a set.
TupleRange	Returns the range of tuples between (and inclusive of) two tuples at the same level.
Union	Returns the union of two input sets.

Metadata-based Set Functions

Functions in this category derive their results using metadata information from the cube.

Function	Result
Ancestors	Returns a set of ancestors up to a specified layer or distance.
Attribute	Returns all base members that are associated with the specified attribute member.
Children	Returns all child members of the input member.
Descendants	Returns the set of descendants of a member at specified layers.
DrilldownByLayer	Drills down members of a set that are at a specified layer.
DrilldownMember	Drills down on any members or tuples of <set1> that are also found in <set2>.
DrillupByLayer	Drills up the members of a set that are below a specified layer.

Function	Result
DrillupMember	Tests two sets for common ancestors, and drills up members in the first set to the layer of the ancestors which are present in the second set.
Extract	Returns a subset containing only the tuples of a specified dimensionality.
Hierarchize	Sorts members according to the default member ordering as represented in the database outline.
LastPeriods	Returns a set of members ending either at the specified member or at the current member in the time dimension.
MemberRange	Returns the range of members positioned between two input members (inclusive) at the same generation or level.
Members	Returns a set of all members of a given dimension, hierarchy, or layer.
PeriodsToDate	Returns a set of dynamic-time-series members from the beginning of a given layer up to a given member in that layer (or up to the default member); or, returns members up to the current member of the Time dimension.
RelMemberRange	Returns a set based on the relative position of the specified member.
Siblings	Returns the siblings of the input member.
Uda	Returns all members that share a specified user-defined attribute.
WithAttr	Returns all base members that are associated with an attribute member of the specified type.
AttributeEx	Given the varying attribute member and the perspective setting, returns the associated base member list.
WithAttrEx	Given the varying attribute dimension, condition, predicate, and perspective setting, returns the base member list satisfying the predicate.
xTD	Functions returning period-to-date values.

Data-based Set Functions

Functions in this category derive their results using data values from the cube.

Function	Result
BottomCount	Returns a set of n elements ordered from smallest to largest, optionally based on an evaluation.
BottomPercent	Returns the smallest possible subset, with elements listed from smallest to largest, of a set for which the total results of a numeric evaluation are at least a given percentage.
BottomSum	Returns the smallest possible subset, with elements listed from smallest to largest, of a set for which the total results of a numeric evaluation are at least a given sum.
Case	Performs conditional expressions.
Filter	Returns those parts of a set which meet the criteria of a search condition.
IIF	Performs a conditional test, and returns an appropriate numeric expression or set depending on whether the test evaluates to true or false.
Leaves	Returns the set of level 0 (leaf) members that contribute to the value of the specified member.

Function	Result
Order	Sorts members of a set in order based on an expression.
TopCount	Returns a set of n elements ordered from largest to smallest, optionally based on an evaluation.
TopPercent	Returns the smallest possible subset, with elements listed from largest to smallest, of a set for which the total results of a numeric evaluation are at least a given percentage.
TopSum	Returns the smallest possible subset, with elements listed from largest to smallest, of a set for which the total results of a numeric evaluation are at least a given sum.

MDX Functions that Return a Tuple

The following functions return a tuple.

Function	Result
CurrentTuple	Returns the current tuple in a set. <i>Current</i> is in the context of query execution mechanics. Use in combination with iterative functions such as Filter .
Item	Extracts a member from a tuple.

MDX Functions that Return a Number

The following functions return a value.

Function	Result
Abs	Returns absolute value of an expression.
Aggregate	Aggregates the Accounts member based on its Time Balance behavior.
Avg	Returns the average of values found in the tuples of a set.
Case	Performs conditional expressions.
CellValue	Returns the numeric value of the current cell.
CoalesceEmpty	Returns the first non #Missing value from the given value expressions.
Count	Returns the count of the number of tuples in a set.
DateDiff	Returns the difference between two input dates.
DatePart	Returns a number representing a date part (such as Week).
EnumText	Returns the text value corresponding to a numeric value in a text list.
EnumValue	Returns the internal numeric value for a text value in a text list.
Exp	Returns the exponent of an expression.

Function	Result
Factorial	Returns the factorial of an expression.
IIF	Performs a conditional test, and returns an appropriate numeric expression or set depending on whether the test evaluates to true or false.
InStr	Returns a number specifying the position of the first occurrence of one string within another.
Int	Returns the next lowest integer value of an expression.
Len	Returns length of a string.
Ln	Returns the natural logarithm of an expression.
Log	Returns the logarithm of an expression to a specified base.
Log10	Returns the base-10 logarithm of an expression.
Max	Returns the maximum of values found in the tuples of a set.
Median	Returns the value of the median tuple of a set.
Min	Returns the minimum of values found in the tuples of a set.
Mod	Returns the modulus (remainder value) of a division operation.
NonEmptyCount	Returns the count of the number of tuples in a set that evaluate to nonempty values.
NTile	Returns a division number of a tuple in a set.
Ordinal	Returns a number indicating depth in the hierarchy.
Percentile	Returns the value of the tuple that is at a given percentile of a set.
Power	Returns the value of the numeric value expression raised to <i>power</i> .
Rank	Returns the numeric position of a tuple in a set.
RealValue	Returns a value for the specified member or tuple without the inherited attribute dimension context.
Remainder	Returns the remainder value of the numeric value expression.
Round	Rounds a numeric value expression to the specified number of digits.
Stddev	Calculates standard deviation based on a sample.
Stddevp	Calculates standard deviation based on a population.
StrToNum	Converts a string to a number.
Sum	Returns the sum of values of tuples in a set.
ToDate	Converts a date string to a value that is usable in calculations.
Truncate	Removes the fractional part of a numeric value expression, returning the integer.

MDX Functions that Return a Dimension

The [Dimension](#) function returns the dimension that contains the input element.

MDX Functions that Return a Layer

The following functions return a [layer](#). A layer is used to group the members of a dimension by hierarchical depth.

In Essbase, a layer is either a generation or a level, indicated by a name or a number.

Function	Result
Generation	Returns the generation of the input member.
Generations	Returns the generation specified by the input numerical depth and the input dimension or hierarchy.
Level	Returns the level of the input member.
Levels	Returns the level specified by the input numerical depth and the input dimension or hierarchy.

MDX Functions that Return a Boolean

The following functions return a Boolean (TRUE or FALSE).

Function	Result
Is	Returns TRUE if two members are identical.
IsAccType	Returns TRUE if the current member has the associated accounts tag.
IsAncestor	Returns TRUE if the first member is an ancestor of the second member.
IsChild	Returns TRUE if the first member is a child of the second member.
IsEmpty	Returns True if the value of an input numeric-value-expression is #MISSING.
IsGeneration	Returns TRUE if the member is in a specified generation.
IsLeaf	Returns TRUE if the member is a level-0 member.
IsLevel	Returns TRUE if the member is in a specified level.
IsSibling	Returns TRUE if the first member is a sibling of the second member.
IsUda	Returns TRUE if the member has the associated UDA tag (user-defined attribute).
IsValid	Returns TRUE if the specified element validates successfully.
Contains	Returns TRUE if a tuple is found within a set.

MDX Functions that Return a Date

The following functions return a date.

Function	Result
DateRoll	To the given date, rolls (adds or subtracts) a number of specific time intervals, returning another date.
GetFirstDate	Returns the start date for a date-hierarchy member.
GetLastDate	Returns the end date for a date-hierarchy member.
GetNextDay	To the given date and the week day, gets the next date after input date that corresponds to the week day.
GetFirstDay	For a given date_part, returns the first day of the time interval for the input date.
GetLastDay	For a given date_part, returns the last day of the time interval for the input date.
ToDateEx	Converts date strings to dates.
Today	Returns a number representing the current date.
JulianDate	For the given UNIX date, gets its Julian date.
UnixDate	For the given Julian date, gets its UNIX date.

MDX Functions that Return a String

The following functions return a string.

Function	Result
FormatDate	Formats date strings.
Concat	Concatenates input strings.
Left	Returns a specified number of characters from the left side of the string.
Right	Returns a specified number of characters from the right side of the string.
LTrim	Trims whitespace on the left of the string.
RTrim	Trims whitespace on the right of the string.
Lower	Converts upper-case string to lower case.
Upper	Converts lower-case string to upper case.
Substring	Returns the substring between a starting and ending position.
NumToStr	Converts a double-precision floating-point value into a decimal string.

MDX Function Reference

Consult the Contents pane for a list of MDX functions by return value.

Abs	Generations	Min
Aggregate	GetFirstDate	Mod
Ancestor	GetFirstDay	NextMember
Ancestors	GetLastDate	NonEmptyCount
Attribute	GetLastDay	NonEmptySubset
AttributeEx	GetNextDay	NTile
Avg	GetRoundDate	NumToStr
BottomCount	Head	OpeningPeriod
BottomPercent	Hierarchize	Order
BottomSum	IIF	Ordinal
Case	InStr	ParallelPeriod
CellValue	Int	Parent
Children	Intersect	Percentile
ClosingPeriod	Is	PeriodsToDate
CoalesceEmpty	IsAccType	Power
Concat	IsAncestor	PrevMember
Contains	IsChild	Rank
Count	IsEmpty	RealValue
Cousin	IsGeneration	RelMemberRange
CrossJoin	IsLeaf	Remainder
CurrentMember	IsLevel	Right
CurrentTuple	IsMatch	Round
DateDiff	IsSibling	RTrim
DatePart	IsUda	Siblings
DateRoll	IsValid	Stddev
DateToMember	Item	Stddevp
DefaultMember	JulianDate	StrToMbr

Abs	Generations	Min
Descendants	Lag	StrToNum
Distinct	LastChild	Subset
Dimension	LastPeriods	Substring
DrilldownByLayer	LastSibling	Sum
DrilldownMember	Lead	Tail
DrillupByLayer	Leaves	Todate
DrillupMember	Left	TodateEx
DTS	Len	Today
EnumText	Level	TopCount
EnumValue	Levels	TopPercent
Except	LinkMember	TopSum
Exp	Ln	Truncate
Extract	Log	TupleRange
Factorial	Log10	Uda
Filter	Lower	Union
FirstChild	LTrim	UnixDate
FirstSibling	Max	Upper
FormatDate	Median	Value
Generate	MemberRange	WithAttr
Generation	Members	WithAttrEx
		xTD

Abs

Returns the absolute value of expression. The absolute value of a number is that number less its sign. A negative number becomes positive, while a positive number remains positive.

Syntax

`Abs (numeric_value_expression)`

Parameter

Description

numeric_value_expression Numeric value expression (see “MDX Grammar Rules” on page 934).

Example

The following example is based on the Demo Basic database. The absolute value is taken in case Variance is a negative number. Absolute Variance is always a non-negative number.

The following query:

```
WITH MEMBER
  [Scenario].[Absolute Variance]
AS
  'Abs([Scenario].[Actual] - [Scenario].[Budget])'
SELECT
  { [Year].[Qtr1].children }
ON COLUMNS,
  { [Scenario].children, [Scenario].[Absolute Variance] }
ON ROWS
FROM
  Demo.Basic
WHERE
  ([Accounts].[Sales], [Product].[VCR], [Market].[San_Francisco])
```

returns the grid:

(axis)	Jan	Feb	Mar
Actual	1323	1290	1234
Budget	1200	1100	1100
Variance	123	190	134
Absolute Variance	123	190	134

Aggregate

Aggregates the Accounts member based on its Time Balance behavior.

Syntax

```
Aggregate ( set [, accounts_member] )
```

Parameter

Description

set

A set containing tuples to be aggregated. If empty, #Missing is returned.

accounts_member

A member from an Accounts dimension. If omitted, the current member from Accounts is used.

Notes

For each tuple in *set*, the value of *accounts_member* is evaluated.

If *accounts_member* has no time balance tag, or if *set* is one-dimensional, this function behaves the same as Sum().

If *accounts_member* has a time balance tag, this function behaves as follows:

- For TB First, returns the value of *accounts_member* for the first tuple in *set*.

- For TB First with SKIP, scans tuples in *set* from first to last and returns first tuple with non-empty value for *accounts_member*.
- For TB Last, returns the value of *accounts_member* for the last tuple in *set*.
- For TB Last with SKIP, scans tuples in *set* from last to first and returns first tuple with non-empty value for *accounts_member*.
- For TB Average, returns the average of values of *accounts_member* at each tuple in *set*.
- For TB Average with SKIP, returns the average of value of *accounts_member* at each tuple in *set* without factoring empty values.

Ancestor

Given the input member, this function returns an ancestor at the specified layer.

Syntax

```
Ancestor ( member , layer | index [, hierarchy ] )
```

Parameter Description

member The member for which an ancestor is sought.

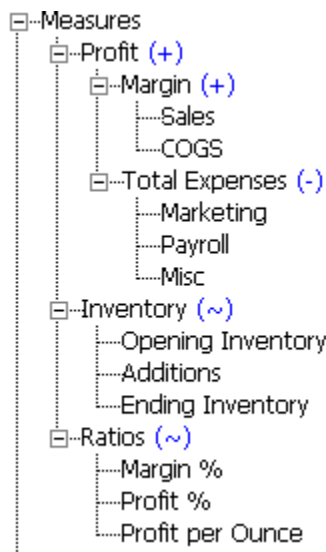
layer Layer specification.

index A number of hierarchical steps up from *member*, locating the ancestor you want returned.

hierarchy Optional. A specific hierarchy within the time dimension.

Notes

- The return value of this function is a member. If you want the return value to be a set, use [Ancestors](#).
- Do not use negative numbers for *index*. If you want to return lower members, use [Descendants](#) instead of Ancestor. `Ancestor([Qtr1], -1)` would return an empty member, not a descendant.
- If you use *layer* to specify a level but no ancestor exists at that level, then the return value is an empty member. For example, in the Sample Basic database, consider the level numbers of the ancestors of the member [Additions] in the [Measures] dimension:



- [Additions], being a leaf-level member, has level number 0.
- [Inventory] has level number 1.
- [Measures] has level number 3, as one of its children [Profit] has level number 2.

The level number of a member = (highest level number among its children) + 1. Therefore, `Ancestor ([Measures].[Additions], [Measures].Levels(2))` returns an empty member, because [Additions] does not have an ancestor with level number 2.

Example

```
Ancestor ( [New York], [Market].levels(2) )
```

returns the member [Market], which is the ancestor of [New York] that is located at level 2 in the outline.

```
Ancestor ([Year].[Jan], [Year].generations(2))
```

returns the member [Qtr1], which is the ancestor of Jan that is located in the second generation of the Year dimension.

```
Ancestor ( [Feb], 2 )
```

returns the member [Year], which is the grandparent of Feb.

```
Ancestor ( [Feb], 0 )
```

returns the member [Feb]. An "ancestor" that is zero steps away is considered to be the member itself.

Ancestors

Given the input member and a layer or distance, this function returns a set of ancestors along with the input member.

When the layer specification is a level, this function returns all ancestors having a level no greater than the input level. For example, `Ancestors ([Additions], [Measures].Levels(2))` returns `{[Inventory], [Additions]}`.

Syntax

```
Ancestors ( member , layer | index )
```

Parameter Description

`member` The member for which a set of ancestors is sought.

`layer` Layer specification.

`index` A number of hierarchical steps up from `member`, locating the highest ancestor you want returned in the result set.

Notes

- Do not use negative numbers for `index`. If you want to return lower members, use [Descendants](#) instead of `Ancestors`. `Ancestors([Qtr1], -1)` would return an empty member, not a descendant.
- If you use `layer` to specify a level but no ancestors exist at that level, then the return value is an empty member.

Example

```
Ancestors ( [New York], [Market].levels(2) )
```

returns `{[Market], [East], [New York]}`, the self-inclusive set of `[New York]` ancestors beginning with the ancestor that is located at level 2 of the `Market` dimension.

```
Ancestors ( [Feb], 1 )
```

returns `{[Qtr1], [Feb]}`, the self-inclusive set of ancestors beginning with the ancestor one step higher than `Feb`.

```
Ancestors ( [Feb], 0 )
```

returns `{[Feb]}`.

Using the `ASOSamp.Sample` database,

```
Ancestors ([94089], [Geography].generations(2))
```

returns `{[West], [CA], [SUNNYVALE - CA], [94089]}`, the self-inclusive set of 94089 ancestors beginning with the second generation of the `Geography` dimension.

Attribute

Returns all base members that are associated with a specified attribute member.

Syntax

```
Attribute ( member )
```

Parameter Description

[member](#) Specification of a member from an attribute dimension.

Example

The following query

```
SELECT
  {[Year].Children}
ON COLUMNS,
  Attribute ([Ounces_12])
ON ROWS
FROM Sample.Basic
```

returns the grid:

(axis)	Qtr1	Qtr2	Qtr3	Qtr4
Cola	5096	5892	6583	5206
Diet Cola	1359	1534	1528	1287
Old Fashioned	1697	1734	1883	1887
Sarsaparilla	1153	1231	1159	1093
Diet Cream	2695	2723	2855	2820

See Also

- [WithAttr](#)

AttributeEx

Returns the set of base members that are associated with a specified varying attribute member or dimension, given the perspective setting.

Syntax

```
AttributeEx ( member|dimension, ANY, tuple|member[,tuple|member] )
```

Parameter	Description
member	Specification of a member from an attribute dimension.
dimension	Specification of an attribute dimension.
ANY	The keyword ANY.

tuple | *member* Level 0 start tuple (or member) of the independent dimension set. The tuple must contain all the discrete dimensions followed by the continuous dimension members, in the same order that the continuous range has been defined.

Parameter	Description
tuple member	Optional level 0 end tuple (or member) of the independent dimension set. The tuple must contain all the discrete dimensions followed by the continuous dimension members, in the same order that the continuous range has been defined.

Example

Consider the following scenario: Products are packaged under different ounces over time and the market state, according to the marketing strategy of the company. Ounces is defined as a varying attribute for the Product dimension, to capture the varying attribute association over the continuous Year dimension and the discrete Market dimension.

Year and Market are the independent dimensions, and level-0 tuple months (for example, Jan) combined with a market state (for example, California) is a perspective for which the varying attribute association is defined.

The following query analyzes the Ounces_32 sales performance of products packaged as Ounces_32 any time from Jul to Dec in New York over all quarters. This is the reality view, which gives the most current view of metrics as they happened over time.

```
WITH PERSPECTIVE REALITY for Ounces
SELECT
  { Qtr1, Qtr2, Qtr3, Qtr4}
ON COLUMNS,
  {AttributeEx(Ounces_32, ANY, ([New York], Jul), ([New York], Dec))}
ON ROWS
FROM
  app.db
WHERE
  (Sales, [New York], Ounces_32);
```

See Also

- [WithAttrEx](#)

Avg

Returns the average of values found in the tuples of a set.

Syntax

```
Avg ( set [,numeric_value_expression [,IncludeEmpty ] ] )
```

Parameter	Description
set	Set specification.
numeric_value_expression	Numeric value expression (see “ MDX Grammar Rules ” on page 934). Avg() sums the numeric value expression and then takes the average.
IncludeEmpty	Use this keyword if you want to include in the average any tuples with #MISSING values. Otherwise, they are omitted by default.

Notes

The average is calculated as (sum over the tuples in the set of *numeric_value_expr*) / *count*, where *count* is the number of tuples in the set. Tuples with missing values are not included in count unless IncludeEmpty is specified.

The return value of Avg is #MISSING if either of the following is true:

- The input set is empty.
- All tuple evaluations result in #MISSING values.

Example

Empty Values Included in Calculation of the Average

The following query

```
WITH MEMBER
  [Market].[Western Avg]
AS
  'Avg ( [Market].[California]:[Market].[Nevada], [Measures].[Sales], INCLUDEEMPTY ) '
SELECT
  { [Product].[Colas].children }
ON COLUMNS,
  { [Market].[West].children, [Market].[Western Avg] }
ON ROWS
FROM
  Sample.Basic
WHERE
  ([Measures].[Sales], [Year].[Jan], [Scenario].[Actual])
```

returns the grid:

(axis)	Cola	Diet Cola	Caffeine Free Cola
California	678	118	145
Oregon	160	140	150
Washington	130	190	#Missing
Utah	130	190	170
Nevada	76	62	#Missing
Western Avg	234.8	140	93

Western Avg for Caffeine Free Cola is 93 because the sales for all Western states is divided by 5, the number of states.

Empty Values Not Included in Calculation of the Average

The following query is the same as the above query, except that it does not use IncludeEmpty:

```
WITH MEMBER
  [Market].[Western Avg]
AS
  'Avg ( [Market].[California]:[Market].[Nevada], [Measures].[Sales] ) '
```

```

SELECT
  { [Product].[Colas].children }
ON COLUMNS,
  { [Market].[West].children, [Market].[Western Avg] }
ON ROWS
FROM
  Sample.Basic
WHERE
  ([Measures].[Sales], [Year].[Jan], [Scenario].[Actual])

```

returning the grid:

(axis)	Cola	Diet Cola	Caffeine Free Cola
California	678	118	145
Oregon	160	140	150
Washington	130	190	#Missing
Utah	130	190	170
Nevada	76	62	#Missing
Western Avg	234.8	140	155

Western Avg for Caffeine Free Cola is 155 because the sales for all Western states is divided by 3, the number of states that do not have empty values for Caffeine Free Cola.

BottomCount

Returns a set of n elements ordered from smallest to largest, optionally based on an evaluation.

This function ignores tuples that resulted in missing values after evaluating *numeric value expression*.

Syntax

```
BottomCount ( set, index [, numeric_value_expression ] )
```

Parameter	Description
<code>set</code>	The set from which the bottom n elements are selected.
<code>index</code>	The number of elements to be included in the set (n).
<code>numeric_value_expression</code>	Optional. An expression further defining the selection criteria (see “MDX Grammar Rules” on page 934).

Example

The following expression

```
Bottomcount ( [Product].levels(0).members, 10, ( [Sales], [Actual] ) )
```

returns the set:

```
{ [200-40], [100-30], [400-30], [300-20], [200-30],
  [100-20], [100-20], [400-20], [400-10], [300-30] }
```

Therefore, the following query

```
SELECT {[Year].levels(1).members} ON COLUMNS,
BottomCount ( [Product].levels(0).members, 10, ( [Sales], [Actual] ) )
ON ROWS
FROM Sample.Basic
WHERE ( [Sales], [Actual] )
```

returns the grid:

(axis)	Qtr1	Qtr2	Qtr3	Qtr4
200-40	2807	2922	2756	3265
100-30	3187	3182	3189	3283
400-30	3763	3962	3995	4041
300-20	4248	4638	4556	4038
200-30	4440	4562	4362	4195
100-20	7276	7957	8057	7179
100-20	7276	7957	8057	7179
400-20	7771	8332	8557	8010
400-10	8614	9061	9527	8957
300-30	8969	9105	9553	9342

See Also

- [TopCount](#)

BottomPercent

Returns the smallest possible subset of a set for which the total results of a numeric evaluation are at least a given percentage. The result set is returned with elements listed from smallest to largest.

Syntax

```
BottomPercent ( set, percentage, numeric_value_expression )
```

Parameter

Description

[set](#)

The set from which the bottom-percentile elements are selected.

percentage

The percentile. This argument must be a value between 0 and 100.

Parameter	Description
numeric_value_expression	The expression that defines the selection criteria (see “ MDX Grammar Rules ” on page 934).

Notes

This function ignores negative and missing values.

Example

The following query returns data for products making up the lowest 5th percentile of all product sales in the Sample Basic database.

```
WITH
  SET [Lowest 5% products] AS
    'BottomPercent (
      { [Product].members },
      5,
      ([Measures].[Sales], [Year].[Qtr2])
    )'

MEMBER
  [Product].[Sum of all lowest prods] AS
  'Sum ( [Lowest 5% products] )'

MEMBER [Product].[Percent that lowest sellers hold of all product sales] AS
  'Sum ( [Lowest 5% products] ) / [Product] '

SELECT
  {[Year].[Qtr2].children}
on columns,
  {
    [Lowest 5% products],
    [Product].[Sum of all lowest prods],
    [Product],
    [Product].[Percent that lowest sellers hold of all product sales]
  }
on rows
FROM Sample.Basic
WHERE ([Measures].[Sales])
```

In the WITH section,

- The named set [Lowest 5% products] consists of those products accounting for the lowest 5 percent of sales in the second quarter. This set includes Birch Beer, Caffeine Free Cola, Strawberry, Sasparilla, and Vanilla Cream.
- The first calculated member, [Product].[Sum of all lowest prods], is used to show the sum of the sales of the products with sales in the lowest fifth percentile.
- The second calculated member, [Product].[Percent that lowest sellers hold of all product sales], is used to show, for each month, how the sales of lowest-selling products compare (as a percentage) to sales of all products in the Product dimension.

This query returns the following grid:

(axis)	Apr	May	Jun
Birch Beer	954	917	1051
Caffeine Free Cola	1049	1065	1068
Strawberry	1314	1332	1316
Sarsaparilla	1509	1552	1501
Vanilla Cream	1493	1533	1612
Sum of all lowest prods	6319	6399	6548
Product	32917	33674	35088
Percent that lowest sellers hold of all product sales	0.192	0.194	0.187

BottomSum

Returns the smallest possible subset of a set for which the total results of a numeric evaluation are at least a given sum. Elements of the result set are listed from smallest to largest.

Syntax

`BottomSum (set, numeric_value_expression, numeric_value_expression)`

Parameter	Description
<code>set</code>	The set from which the lowest-summing elements are selected.
<code>numeric_value_expression1</code>	The given sum (see “ MDX Grammar Rules ” on page 934).
<code>numeric_value_expression2</code>	The numeric evaluation (see “ MDX Grammar Rules ” on page 934).

Notes

- If the total results of the numeric evaluation do not add up to the given sum, an empty set is returned.
- This function ignores negative and missing values.

Example

The following query selects Qtr1 and Qtr2 sales for the lowest selling products in Qtr1 (where Sales totals at least 10000).

```
SELECT
  {[Year].[Qtr1], [Year].[Qtr2]}
ON COLUMNS,
  {
    BottomSum(
      [Product].Members, 10000, [Year].[Qtr1]
    )
  }
ON ROWS
```

```
FROM Sample.Basic
WHERE ([Measures].[Sales])
```

This query returns the grid:

(axis)	Qtr1	Qtr2
200-40	2807	2922
100-30	3187	3182
400-30	3763	3962
300-20	4248	4638

Case

The CASE keyword begins a conditional expression. There are two types of conditional test you can perform using CASE: simple case expression and searched case expression.

Syntax

The simple case expression evaluates *case_operand* and returns a result based on its value, as specified by WHEN or ELSE clauses. The result of a case expression can be a value expression or a set. If no ELSE clause is specified, and none of the WHEN clauses is matched, an empty value/empty set is returned.

```
CASE
case_operand
simple_when_clause...
[ else_clause ]
END
```

In searched case expression, each WHEN clause specifies a search condition and a *result* to be returned if that search condition is satisfied. The WHEN clauses are evaluated in the order specified. The result is returned from the first WHEN clause in which the search condition evaluates to TRUE. The result can be a value expression or a set. If no ELSE clause is specified, and none of the search conditions in the WHEN clauses evaluate to TRUE, an empty value/empty set is returned.

```
CASE
searched_when_clause...
[ else_clause ]
END
```

Parameter	Description
case_operand	An expression to evaluate.
simple_when_clause	One or more WHEN/THEN statements. Syntax: WHEN <i>when_operand</i> THEN <i>result</i> <ul style="list-style-type: none"> ● <i>when_operand</i>: A value expression. ● <i>result</i>: A numeric value expression, a string value expression, or a set.

Parameter	Description
else_clause	Optional. Syntax: <code>ELSE numeric_value_expression set string_value_expression</code>
searched_when_clause	One or more WHEN/THEN statements. Syntax: <code>WHEN search_condition THEN result</code> <ul style="list-style-type: none"> • <i>search_condition</i>: A value expression. • <i>result</i>: A numeric value expression, a string value expression, or a set.

Example

Example for Simple Case Expression

In the following query, the calculated member `[Measures].[ProductOunces]` is evaluated based on the value of the Ounce attribute for the current member of the Product dimension.

```
WITH MEMBER [Measures].[ProductOunces] AS
'Case Product.CurrentMember.Ounces
    when 32 then 32
    when 20 then 20
    when 16 then 16
    when 12 then 12
    else 0
end'
SELECT
{ [Measures].[ProductOunces] } ON COLUMNS,
{ [Product].Members } ON ROWS
FROM Sample.Basic
```

This query returns the following result:

(axis)	ProductOunces
Product	0
Colas	0
Cola	12
Diet Cola	12
Caffeine Free Cola	16
Root Beer	0
Old Fashioned	12
Diet Root Beer	16
Sarsaparilla	12
Birch Beer	16
Cream Soda	0

(axis)	ProductOunces
Dark Cream	20
Vanilla Cream	20
Diet Cream	12
Fruit Soda	0
Grape	32
Orange	32
Strawberry	32
Diet Drinks	0
Diet Cola	0
Diet Root Beer	0
Diet Cream	0

Example for Searched Case Expression

The following query divides products into different profit categories based on Profit, and returns categories for each product.

```
WITH MEMBER [Measures].[ProfitCategory] AS
' Case
    when Profit > 10000 then 4
    when Profit > 5000 then 3
    when Profit > 3000 then 2
    else 1
end'
SELECT
{ [Measures].[ProfitCategory] } ON COLUMNS,
{ [Product].Members } ON ROWS
FROM Sample.Basic
```

This query returns the following result:

(axis)	ProfitCategory
Product	4
Colas	4
Cola	4
Diet Cola	3
Caffeine Free Cola	1
Root Beer	4

(axis)	ProfitCategory
Old Fashioned	3
Diet Root Beer	4
Sarsaparilla	2
Birch Beer	2
Cream Soda	4
Dark Cream	4
Vanilla Cream	1
Diet Cream	4
Fruit Soda	4
Grape	4
Orange	3
Strawberry	1
Diet Drinks	4
Diet Cola	3
Diet Root Beer	4
Diet Cream	4

CellValue

Returns the numeric value of the current cell.

Syntax

CellValue

Notes

- This function can be useful when defining format strings for a member. Most MDX expressions can be used to specify format strings; however, format strings cannot contain references to values of data cells other than the current cell value being formatted. Use this function to reference the current cell value.
- Enclose all format strings within the MdxFormat() directive as shown in the examples.

Example

Example 1

The following format string displays negative values for the current measure if the current [AccountTypes] member is of type “Expense”. CellValue refers to the current cell value that is being formatted. The CurrentMember function in the expression refers to the context of the cell being formatted.

```
/* Display negative values if current Account is an Expense type account */
MdxFormat (
  IIF(IsUda(AccountTypes.CurrentMember, "Expense"),
      NumToStr(-CellValue()),
      NumToStr(CellValue()))
)
```

Example 2

The following format string displays negative cell values as positive values enclosed in parentheses.

```
MdxFormat (
  IIF (
    CellValue() < 0,
    Concat(Concat("(", numtostr(-CellValue())), ")" ),
    numtostr(CellValue())
  )
)
```

Example 3

This example illustrates a dynamic member [Variance %] along the [Scenario] dimension. [Variance %] has the following formula, which specifies how to calculate its value from [Actual] and [Budget].

[Variance %] Formula

```
IIF(Is(Measures.CurrentMember, Title) OR
    Is(Measures.CurrentMember, Performance),
    (Actual - Budget) * 10, (Actual - Budget)*100/Budget)
```

[Variance %] also has the following format string, which specifies how its values should be displayed. In this case, based on the percentage value computed for a [Variance %] cell, a text value is displayed which conveys the importance of the number.

[Variance %] Format String

```
MdxFormat (
CASE
  WHEN CellValue() <= 5 THEN          "Low"
  WHEN CellValue() <= 10 THEN       "Medium"
  WHEN CellValue() <= 15 THEN       "High"
  ELSE                               "Very High"
```

```
END
)
```

Children

Returns a set of all child members of the specified member.

Syntax

```
member.Children
```

```
Children ( member )
```

Parameter Description

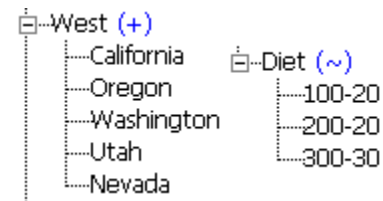
member A member specification.

Notes

If the input member does not have any children (is a level-0 member), this function returns an empty set.

Example

This example uses the following parts of the Sample Basic outline:



The following expression

```
([West].children)
```

returns the set:

```
{ [California], [Oregon], [Washington], [Utah], [Nevada] }
```

And the following expression

```
([Diet].children)
```

returns the set:

```
{ [100-20], [200-20], [300-30] }
```

Therefore, the following query

```
SELECT
  {[West].children}
ON COLUMNS,
  {[Diet].children}
ON ROWS
FROM Sample.Basic
```

returns the grid:

(axis)	California	Oregon	Washington	Utah	Nevada
100-20	-1587	338	231	398	86
200-20	2685	1086	579	496	167
300-30	1328	288	1217	413	362

ClosingPeriod

Returns the last descendant of a layer, or the last child of the Time dimension.

Syntax

```
ClosingPeriod ( [ layer [, member ] ] )
```

Parameter Description

layer Layer specification.

member Optional member specification. If omitted, the last child of the Time dimension is assumed (for example, Qtr4 in Sample Basic).

Notes

The return value of this function varies depending on the input.

1. When both *layer* and *member* arguments are given as input, Closingperiod returns the last descendant of the input member at the input layer. For example, `Closingperiod(Year.generations(3), Qtr3)` returns Sep. If the input *member* and *layer* are the same layer, the output is the input member. For example, `Closingperiod(Year.generations(3), Sep)` returns Sep.
2. When only the *layer* argument is specified, the input member is assumed to be the current member of the dimension used in the layer argument. Closingperiod returns the last descendant of that dimension, at the input layer. For example, `Closingperiod(Year.generations(3))` returns Dec.
3. When no arguments are specified, the input member is assumed to be the current member of the Time dimension, and ClosingPeriod returns the last child of that member. Do not use this function without arguments if there is no dimension tagged as Time.

Example

The following query

```
WITH
MEMBER [Measures].[Starting Inventory] AS
,
IIF (
  IsLeaf (Year.CurrentMember) ,
  [Measures].[Opening Inventory] ,
  ([Measures].[Opening Inventory] ,
  OpeningPeriod (
    [Year].Levels(0) ,
```



```

        [Year].CurrentMember
    )
)
)'
MEMBER [Measures].[Closing Inventory] AS
'
IIF (
    Isleaf(Year.CurrentMember),
    [Measures].[Ending Inventory],
    ([Measures].[Closing Inventory],
    ClosingPeriod (
        [Year].Levels(0),
        [Year].CurrentMember
    )
)
)
)'
SELECT
CrossJoin (
    { [100-10] },
    { [Measures].[Starting Inventory], [Measures].[Closing Inventory] }
)
ON COLUMNS,
Hierarchize ( [Year].Members , POST)
ON ROWS
FROM Sample.Basic

```

returns the grid:

(axis)	100-10	100-10
(axis)	Starting Inventory	Closing Inventory
Jan	14587	14039
Feb	14039	13566
Mar	13566	13660
Qtr1	14587	13660
Apr	13660	14172
May	14172	15127
Jun	15127	15580
Qtr2	13660	15580
Jul	15580	14819
Aug	14819	14055
Sep	14055	13424
Qtr3	15580	13424

(axis)	100-10	100-10
(axis)	Starting Inventory	Closing Inventory
Oct	13424	13323
Nov	13323	13460
Dec	13460	12915
Qtr4	13424	12915
Year	14587	12915

See Also

- [OpeningPeriod](#)
- [LastPeriods](#)
- [ParallelPeriod](#)
- [PeriodsToDate](#)

CoalesceEmpty

Returns the first (from the left) non #Missing value from the given value expressions.

Syntax

`CoalesceEmpty (numeric_value_expression1, numeric_value_expression2)`

Parameter

Description

`numeric_value_expression1` A numeric value expression (see “[MDX Grammar Rules](#)” on page 934).

`numeric_value_expression2` A numeric value expression (see “[MDX Grammar Rules](#)” on page 934).

Notes

This function returns *numeric_value_expression2* if *numeric_value_expression1* is #MISSING; otherwise it returns *numeric_value_expression1*.

Example

```
CoalesceEmpty([Profit per Ounce], 0)
```

returns the [Profit per Ounce] value if it is not #MISSING; returns zero otherwise. This can be used inside the Order function to coalesce all #MISSING values to zero, as shown in the next example:

```
Order([Product].Members, CoalesceEmpty([Profit per Ounce], 0))
```

Without CoalesceEmpty in the value expression, the Order function would skip all [Product] members with MISSING values for [Profit per Ounce].

See Also

- [Order](#)

Concat

Returns the concatenated input strings.

Syntax

```
Concat ( string [, string +] )
```

Parameter Description

string A string.

string + Optional. A second string, or a list of multiple additional strings. If omitted, this function returns the single input string.

Example

```
Concat ("01", "01")
```

Contains

Returns TRUE if a tuple is found within a set; otherwise returns FALSE.

Syntax

```
Contains ( member_or_tuple, set )
```

Parameter Description

member_or_tuple A [member](#) or a [tuple](#).

set The set to search.

Example

The following expression returns TRUE.

```
Contains ([Oregon], { [California], [Oregon] })
```

Count

Returns the number of tuples in a set (the cardinality of the set). This function counts all tuples of the set regardless of empty values. If you wish to count only tuples that evaluate to nonempty values, use [NonEmptyCount](#).

Syntax

```
Count ( set [, IncludeEmpty] )
```

Parameter Description

set The set for which a tuple count is needed.

IncludeEmpty Optional and default (empty values are counted even if this keyword is omitted).

Notes

This function returns a zero if the input set is empty.

Example

```
WITH MEMBER
  [Measures].[Prod Count]
AS
  'Count (
    Crossjoin (
      {[Measures].[Sales]},
      {[Product].children}
    )
  )'
SELECT
  { [Scenario].[Actual], [Scenario].[Budget] }
ON COLUMNS,
  {
    Crossjoin (
      {[Measures].[Sales]},
      {[Product].children}
    ),
    ([Measures].[Prod Count], [Product])
  }
ON ROWS
FROM
  Sample.Basic
WHERE
  ([Year].[Jan], [Market].[New York])
```

returns the grid:

(axis)		Actual	Budget
Sales	Colas	678	640
	Root Beer	551	530
	Cream Soda	663	510
	Fruit Soda	587	620
	Diet Drinks	#Missing	#Missing
Prod Count	Product	5	5

The WITH section of the query calculates the count of all products for which a data value exists. The SELECT section arranges the members shown on columns and rows. The entire query is sliced by January and New York in the WHERE section; though those members are not shown in the grid, the data is applicable to those members.

Cousin

Returns a child member at the same position as a member from another ancestor.

Syntax

Cousin (*member1*, *member2*)

Parameter Description

member1 A child member. For example, [Year] . [Qtr1].

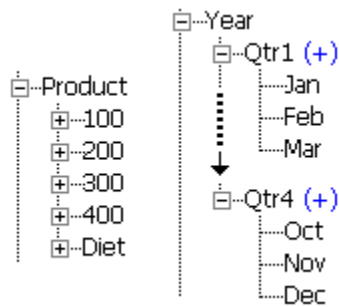
member2 An ancestor for which Cousin() should the return child member at the same position as *member1*.

Notes

Assuming a symmetric hierarchy, Cousin takes as input one member (*member1*) from one hierarchy and an ancestor member (*member2*) of another hierarchy, and returns the child of *member2* that is at the same position as *member1*.

Example

This example uses the following parts of the Sample Basic outline:



The following expression

```
{ Cousin ( [Qtr2].[Apr], [Qtr4] ) }
```

returns the member:

```
[Qtr4].[Oct]
```

And the following expression

```
[Product].generations(2).members
```

returns the set:

```
{ [100], [200], [300], [400], [Diet] }
```

Therefore, the following query

```
SELECT
  { Cousin ( [Qtr2].[Apr], [Qtr4] ) }
ON COLUMNS,
  [Product].generations(2).members
ON ROWS
FROM Sample.Basic
```

returns the grid:

(axis)	Oct
100	2317
200	2505
300	2041
400	1790
Diet	2379

CrossJoin

Returns the cross-product of two sets from different dimensions.

Syntax

```
CrossJoin ( set1, set2 )
```

Parameter Description

- `set1` A set to cross with `set2`.
- `set2` A set to cross with `set1`. Must not include any dimension used in `set1`.

Notes

This function returns the cross-product of two sets from different dimensions. If the two sets share a common dimension, an error is returned.

If one of the input sets is empty, the output set will be empty as well. For example, the output will be empty if the input set is `[Root Beer].children` but `[Root Beer]` has no children.

The order of the sets (and their constituent tuples) provided to the `CrossJoin` function have an effect on the order of the tuples in the result set. For example,

```
CrossJoin({a, b}, {c, d})
```

```
returns {(a, c), (a, d), (b, c), (b, d)}
```

```
CrossJoin({a, b, c}, {d, e, f})
```

```
returns {(a, d), (a, e), (a, f), (b, d), (b, e), (b, f), (c, d), (c, e),
(c, f)}
```

Be aware of the order of the output set when using the results of `CrossJoin` with other order-dependent set functions; for example, [Head](#) or [Tail](#).

Example

Example 1

The following expression

```
CrossJoin({[Qtr1], [Qtr2]}, {[New York], [California]})
```

returns the set:

```
{([Qtr1], [New York]), ([Qtr1], [California]),  
 ([Qtr2], [New York]), ([Qtr2], [California])}
```

Therefore, the following query

```
SELECT  
CrossJoin({[Qtr1], [Qtr2]}, {[New York], [California]})  
ON COLUMNS  
FROM sample.basic
```

returns the grid:

Qtr1	Qtr1	Qtr2	Qtr2
New York	California	New York	California
1656	3129	2363	3288

Example 2

The following expression

```
CrossJoin({[Qtr1], [Qtr2], [Qtr3]}, {[New York], [California], [Texas]})
```

returns the set

```
{([Qtr1], [New York]), ([Qtr1], [California]), ([Qtr1], [Texas]),  
 ([Qtr2], [New York]), ([Qtr2], [California]), ([Qtr2], [Texas]),  
 ([Qtr3], [New York]), ([Qtr3], [California]), ([Qtr3], [Texas])}
```

Therefore, the following query

```
SELECT  
CrossJoin({[Qtr1], [Qtr2], [Qtr3]}, {[New York], [California], [Texas]})  
ON AXIS(0)  
FROM Sample.Basic
```

returns the grid:

Qtr1	Qtr1	Qtr1	Qtr2	Qtr2	Qtr2	Qtr3	Qtr3	Qtr3
New York	California	Texas	New York	California	Texas	New York	California	Texas
1656	3129	1582	2363	3288	1610	1943	3593	1703

Example 3

The following expression

```
CrossJoin ([100].children, [Profit].children)
```

returns the set:

```
{([100-10], Margin), ([100-10], [Total Expenses]),
 ([100-20], Margin), ([100-20], [Total Expenses]),
 ([100-30], Margin), ([100-30], [Total Expenses])}
```

Therefore, the following query

```
SELECT
  {[Market].levels(1).members}
ON COLUMNS,
  CrossJoin ([100].children, [Profit].children)
ON ROWS
FROM Sample.Basic
```

returns the grid:

(axis)	(axis)	East	West	South	Central
100-10	Margin	15762	8803	5937	8124
	Total Expenses	4633	4210	2361	4645
100-20	Margin	1785	3707	2767	7426
	Total Expenses	671	4241	1570	3495
100-30	Margin	871	1629	#Missing	3975
	Total Expenses	458	2139	#Missing	1895

CurrentMember

Returns the current member in the input dimension.

The current member is evaluated in the context of query execution mechanics. Used in conjunction with iterative functions such as [Filter](#), at every stage of iteration the member being operated upon is the current member.

Syntax

```
dimension.CurrentMember
```

```
CurrentMember ( dimension )
```

Parameter Description

dimension A dimension specification.

Notes

This function returns the child of an implied shared member instead of the member itself. To avoid this behavior when using CurrentMember in MDX formulas and calculated members, tag the parent with the "Never Share" property.

An implied share occurs when a parent has only one child, or only one child that consolidates. For more information, see "Understanding Shared Members" in the *Oracle Essbase Database Administrator's Guide*.

Example

The following query selects the quarters during which sales growth is 3% or more compared to the previous month.

```
SELECT
Filter (
  [Year].Children, -- outer loop
  Max (
    Except (
      [Year].CurrentMember.Children, -- current in outer loop
      { [Year].[Jan] }
    ),
    ( [Year].CurrentMember -- current in Max loop
      / [Year].CurrentMember.PrevMember)
  ) >= 1.03
)
ON axis(0)
FROM Sample.Basic
WHERE ([Measures].[Sales])
```

Returns the grid:

Qtr2	Qtr4
101679	98141

CurrentTuple

Returns the current tuple in a set. *Current* is in the context of query execution mechanics. Use in combination with iterative functions such as Filter.

Syntax

```
CurrentTuple ( set )
```

```
set.Current
```

```
set.CurrentTuple
```

Parameter Description

set A set specification. This argument should be a named set, defined in the [WITH section](#).

Example

The following example finds all Product, Market combinations for which Sales data exists.

```
WITH SET [NewSet]
AS 'CrossJoin([Product].Children, [Market].Children)'
SELECT
  Filter([NewSet], NOT IsEmpty([NewSet].CurrentTuple))
```

```

ON COLUMNS
FROM Sample.Basic
WHERE
    {[Sales]}

```

This query returns the following grid:

100				200	...	400	Diet			
East	West	South	Central	East	...	Central	East	West	South	Central
27740	28306	16280	33808	23672	...	33451	7919	36423	18676	42660

DateDiff

Returns the difference (number) between two input dates in terms of the specified date-parts, following a standard Gregorian calendar.

Syntax

```
DateDiff ( date1, date2, date_part )
```

Parameter Description

date1 A number representing the input date between January 1, 1970 and Dec 31, 2037. The number is the number of seconds elapsed since midnight, January 1, 1970. To retrieve this number, use any of the following functions: Today(), TodateEx(), GetFirstDate(), GetLastDate(), DateRoll().

Date-time attribute properties of a member can also be used to retrieve this number. For example,

- Product.currentmember.[Intro Date] returns the product introduction date for the current product in context.
- [Cola].[Intro Date] returns the product introduction date for Cola.

date2 A second input date. See *date1*.

date_part Defined time components as per the standard calendar.

- DP_YEAR - Year of the input date.
- DP_QUARTER - Quarter of the input date.
- DP_MONTH - Month of the input date.
- DP_WEEK - Week of the input date.
- DP_DAY - Day of the input date.

Notes

Based on the input *date_part*, the difference between the two input dates is counted in terms of time component specified.

Example: For input dates June 14, 2005 and Oct 10, 2006,

- DP_YEAR returns the difference in the year component. (2006 - 2005 = 1)
- DP_QUARTER returns the distance between the quarters capturing the input dates. (Quarter 4, 2006 - Quarter 2, 2005 = 6)

- DP_MONTH returns the distance between the months capturing the input dates. (Oct 2006 - June 2005 = 16)
- DP_WEEK returns the distance between the weeks capturing the input dates. Each Standard calendar week is defined to start on Sunday and it spans 7 days. (Oct 10, 2006 - June 14, 2005 = 69)
- DP_DAY returns the difference between the input dates in terms of days. (483 days)

Example

The following query returns weekly sales for the last 6 months for the product Cola in the market California.

```
SELECT
{sales} ON COLUMNS,
Filter(
[Time dimension].Weeks.members,
Datediff(
    GetFirstDate([Time dimension].CurrentMember),
    Today(),
    DP_MONTH
) < 6
)
ON ROWS
FROM Mysamp.Basic
WHERE (Actual, California, Cola);
```

DatePart

This function returns the Year/Quarter/Month/Week/Weekday/DayOfYear/Day as a number, given the input date and a date part, following the standard Gregorian calendar.

Syntax

```
DatePart ( date, date_part_ex )
```

Parameter	Description
-----------	-------------

date	<p>A number representing the input date between January 1, 1970 and Dec 31, 2037. The number is the number of seconds elapsed since midnight, January 1, 1970. To retrieve this number, use any of the following functions: Today(), TodateEx(), GetFirstDate(), GetLastDate(), DateRo</p> <p>Date-time attribute properties of a member can also be used to retrieve this number. For example,</p> <ul style="list-style-type: none"> ● Product.currentmember.[Intro Date] returns the product introduction date for the current product in context. ● [Cola].[Intro Date] returns the product introduction date for Cola.
------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Parameter	Description
-----------	-------------

date_part_ex Defined time components as per the standard calendar.

- DP_YEAR - Year of the input date, in `yyyy` format.
- DP_QUARTER - Quarter of the year (1 to 4) for the input date.
- DP_MONTH - Month of the year (1 to 12) for the input date.
- DP_WEEK - Week of the year for the input date (1 to 54).
- DP_WEEKDAY - Week day of the input date. (1 - Sunday, 2 - Monday, ... 7 - Saturday).
- DP_DAYOFYEAR - Day of the year numbering (1 to 366).
- DP_DAY - Day of the month for the input date (1 to 31).

Notes

Based on the requested time component, the output is as follows:

- DP_YEAR returns the year of the input date in `yyyy` format.
- DP_QUARTER returns the quarter of the year (1 to 4) for the input date.
- DP_MONTH returns the month of the year (1 to 12) for the input date.
- DP_WEEK returns the week of the year for the input date (1 to 54).
- DP_WEEKDAY returns the week day of the input date. (1 - Sunday, 2 - Monday, ... 7 - Saturday).
- DP_DAYOFYEAR returns the day of the year numbering (1 to 366).
- DP_DAY returns the day of the month for the input date (1 to 31).

Example: For June 14, 2005,

DP_YEAR returns 2005 (the year member, in `yyyy` format).

DP_QUARTER returns 2 (Second quarter of the year)

DP_MONTH returns 6 (Sixth month of the year)

DP_WEEK returns 24 (24th week of the year)

DP_WEEKDAY returns 4 (for Wednesday. Sunday = 1)

DP_DAYOFYEAR returns 165 (165th day of the year)

DP_DAY returns 14 (14th day of the month)

Example

The following query returns the quarterly sales for the second quarter across all years for the product Cola in the market California.

```
SELECT
  {[Sales]}
  ON COLUMNS,
  {
    Filter(
      [Time dimension].Quarters.members,
      Datepart(
```

```

        getFirstDate([Time dimension].CurrentMember),
        DP_QUARTER
    ) = 2
)
}
    ON ROWS,
FROM MySamp.Basic
WHERE (Actual, Cola, California);

```

DateRoll

To the given date, rolls (adds or subtracts) a number of specific time intervals, returning another date. This function assumes a standard Gregorian calendar.

Syntax

```
DateRoll ( date, date_part, number )
```

Parameter Description

date A number representing the date between January 1, 1970 and Dec 31, 2037. The number is the number of seconds elapsed since midnight, January 1, 1970. To retrieve this number, use any of the following functions: Today(), TodayEx(), GetFirstDate(), GetLastDate().

Date-time attribute properties of a member can also be used to retrieve this number. For example,

- `Product.currentmember.[Intro Date]` returns the product introduction date for the current product in context.
- `[Cola].[Intro Date]` returns the product introduction date for Cola.

date_part Defined time components as per the standard calendar.

- DP_YEAR - Year of the input date.
- DP_QUARTER - Quarter of the input date.
- DP_MONTH - Month of the input date.
- DP_WEEK - Week of the input date.
- DP_DAY - Day of the input date.

number Number of time intervals to add or subtract.

Notes

Based on input *date_part* and dateroll *number*, the date is moved forward or backward in time.

Example: For input date June 14, 2005 and input dateroll number 5,

- DP_YEAR adds 5 years to the input date. (June 14, 2010)
- DP_QUARTER adds 5 quarters to the input date. (June 14, 2005 + 5 quarters = June 14, 2005 + 15 months = Sept 14, 2006)
- DP_MONTH adds 5 months to the input date (June 14, 2005 + 5 months = Nov 14, 2005)
- DP_WEEK adds 5 weeks to the input date (June 14, 2005 + 5 weeks = June 14, 2005 + 35 days = July 19, 2005)
- DP_DAY adds 5 days to the input date. (June 14, 2005 + 5 days = June 19, 2005)

Example

The following query returns actual weekly sales, rolling back for six months from Apr 2005 (inclusive), for the product Cola in the market California.

```
SELECT
  {[Sales]}
ON COLUMNS,
  {DateToMember
    (
      DateRoll(
        GetFirstDate ([Apr 2005]),
          DP_MONTH,
          6
        ),
      [Time dimension].Dimension,
      [Time dimension].[WEEKS]
    ): ClosingPeriod([Time dimension].[Weeks], [Apr 2005])
  } ON ROWS
FROM MySamp.Basic
Where (Actual, California, Cola);
```

DateToMember

Returns the date-time dimension member specified by the input date and the input layer.

Syntax

```
DateToMember ( date, dimension [,layer])
```

Parameter Description

date A number representing the input date between January 1, 1970 and Dec 31, 2037. The number is the number of seconds elapsed since midnight, January 1, 1970. To retrieve this number, use any of the following functions: Today(), TodateEx(), GetFirstDate(), GetLastDate(), DateRoll().

Date-time attribute properties of a member can also be used to retrieve this number. For example,

- `Product.currentmember.[Intro Date]` returns the product introduction date for the current product in context.
- `[Cola].[Intro Date]` returns the product introduction date for Cola.

dimension A date-time dimension specification.

layer Optional. A date-time dimension layer specification. If not specified, defaults to the date-time dimension's leaf generation.

Notes

- This function is applicable only to aggregate storage databases.
- This function is only applicable if there is a date-time dimension in the outline.

Example

Consider the following Time-Date dimension hierarchy:

```

Time dimension (gen 1)
  Years (gen 2)
    Semesters (gen 3)
      Quarters (gen 4)
        Months (gen 5)
          Weeks (gen 6)
            Days (gen 7)

```

The following query returns sales for the week containing Dec 25, 2006 for the product Cola in the market California.

```

SELECT
{Sales} ON COLUMNS,
{
DateToMember(
  TodateEx("Mon dd yyyy", "December 25 2006"),
  [Time dimension].Dimension,
  [Time dimension].[Weeks])
} ON ROWS
FROM MySamp.Basic
WHERE (Actual, California, Cola);

```

DefaultMember

Returns the default member in the input dimension. In Essbase, the top member of the input dimension is returned.

Syntax

```

dimension.DefaultMember

DefaultMember ( dimension )

```

Parameter Description

dimension A dimension specification.

Example

```

DefaultMember ( [Market] )

returns the member [Market].

DefaultMember ( [Florida].Dimension )

returns the member [Market].

DefaultMember ( [Bottle] )

returns the member [Pkg Type].

```

Descendants

Returns the set of descendants of a member at a specified level or distance, optionally including or excluding descendants in other levels. The members are returned in hierarchized order; for example, parent members are followed by child members.

Syntax

```
Descendants ( member , [{ layer | index }[, Desc_flags ]])
```

Parameter Description

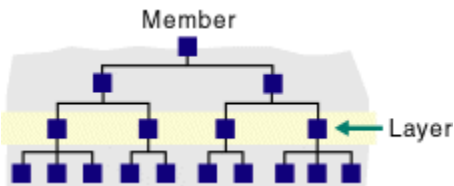
- member** The member for which descendants are sought.
- layer** Optional. Layer specification indicating the depth of the descendants to return.
- index** Optional. A number of hierarchical steps down from *member*, locating the descendants you want returned.
- Desc_flags** Optional. Keywords which further indicate which members to return. These keywords are available only if *layer* or *index* is specified.
See [Values for Desc_flags](#)

Notes

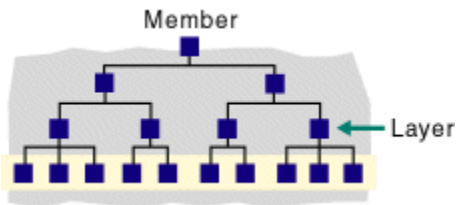
Values for Desc_flags

For all flags, SELF refers to *layer*; therefore, BEFORE indicates "before the layer" and AFTER indicates "after the layer."

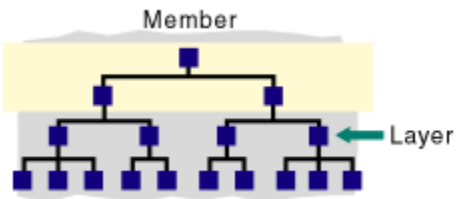
- SELF—Include only members in *layer*, including *member* only if *member* is in *layer*.



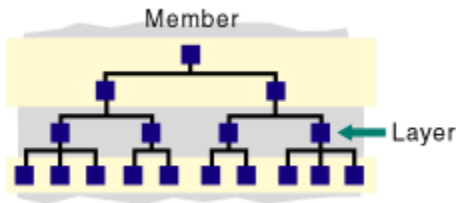
- AFTER—Include members below *layer*, but not the members of *layer*.



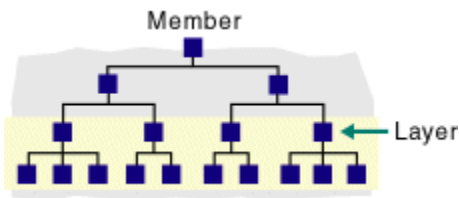
- BEFORE—Include *member* and all its descendants that are higher in the hierarchy than *layer*, excluding *layer* and anything below it.



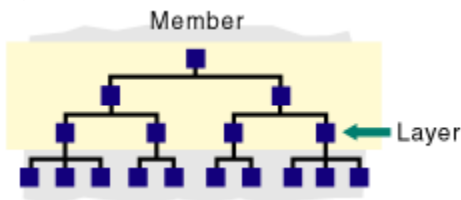
- BEFORE_AND_AFTER—Include *member* and all its descendants, down to level 0, but excluding members in *layer*.



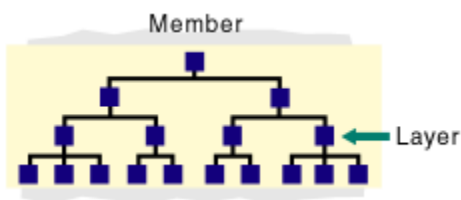
- SELF_AND_AFTER—Include members in *layer* and all descendants below *layer*.



- SELF_AND_BEFORE—Include *member* and all its descendants, down to and including *layer*.



- SELF_BEFORE_AFTER—Include *member* and all its descendants.



- LEAVES—Include only level-0 descendants between *member* and *layer*.

Example

The following query

```
SELECT
  Descendants ( [Year] )
ON COLUMNS
FROM sample.basic
```

returns the grid:

Year	Qtr1	Jan	Feb	Mar	Qtr2	Apr	May	Jun	Qtr3	Jul	Aug	Sep	Qtr4	Oct	Nov	Dec
12656	2747	924	888	935	3352	1011	1071	1270	3740	1334	1304	1102	2817	907	884	1026

The following expressions return the following sets

```
Descendants ( [Year], 2 )
```

returns { ([Jan] : [Dec]) }, which is the range of members found two steps below Year.

```
Descendants ( [Year], 2, BEFORE )
```

returns { [Year], [Qtr1], [Qtr2], [Qtr3], [Qtr4] }, which is the set of Year and its descendants that occur BEFORE the layer that is two steps below Year.

```
Descendants ( [Market], [West].level )
```

returns { [East], [West], [South], [Central] }, which is the set of Market's descendants found at the level of West.

```
Descendants([Market])
```

is equivalent to Descendants([Market], [Market].level, SELF_BEFORE_AFTER). It returns all descendants of Market:

```
{ [Market],  
  [East], [New York], [Massachusetts], [Florida], [Connecticut], [New Hampshire],  
  [West], [California], [Oregon], [Washington], [Utah], [Nevada],  
  [South], [Texas], [Oklahoma], [Louisiana], [New Mexico],  
  [Central], [Illinois], [Ohio], [Wisconsin], [Missouri], [Iowa], [Colorado] }
```

```
Descendants([Market], [Region])
```

is equivalent to Descendants([Market], [Region], SELF), where [Region] is an alias. It returns all members at [Region] level:

```
{ [East], [West], [South], [Central] }
```

```
Descendants([Market], [State], SELF)
```

returns all descendants of [Market] at [State] level:

```
{ [New York], [Massachusetts], [Florida], [Connecticut], [New Hampshire],  
  [California], [Oregon], [Washington], [Utah], [Nevada], [Texas],  
  [Oklahoma], [Louisiana], [New Mexico], [Illinois], [Ohio], [Wisconsin],  
  [Missouri], [Iowa], [Colorado] }
```

```
Descendants([Market], [State], BEFORE)
```

returns all regions and [Market]:

```
{ [Market], [East], [West], [South], [Central] }
```

```
Descendants([Market], [State], AFTER)
```

returns an empty set, because there are no levels below [State] level in the [Market] dimension:

```
{ }
```

```
Descendants([Market], [Region], AFTER)
```

returns all states in the [Market] dimension:

```
{ [New York], [Massachusetts], [Florida], [Connecticut], [New Hampshire],  
  [California], [Oregon], [Washington], [Utah], [Nevada], [Texas],  
  [Oklahoma], [Louisiana], [New Mexico], [Illinois], [Ohio], [Wisconsin],  
  [Missouri], [Iowa], [Colorado] }
```

```
Descendants([Market], [State], LEAVES)
```

returns all level-0 members between [Market] level and [State] level, including both levels:

```
{[New York], [Massachusetts], [Florida], [Connecticut], [New Hampshire],  
 [California], [Oregon], [Washington], [Utah], [Nevada], [Texas],  
 [Oklahoma], [Louisiana], [New Mexico], [Illinois], [Ohio], [Wisconsin],  
 [Missouri], [Iowa], [Colorado]}
```

```
Descendants([Market], 1)
```

The second argument specifies a distance of 1 from [Market] level, which is [Region] level. So this expression is equivalent to Descendants([Market], [Region]). It returns:

```
{[East], [West], [South], [Central]}
```

```
Descendants([Market], 2, SELF_BEFORE_AFTER)
```

is equivalent to Descendants([Market], [State], SELF_BEFORE_AFTER). It returns:

```
{[Market],  
 [East], [New York], [Massachusetts], [Florida], [Connecticut], [New Hampshire]  
 [West], [California], [Oregon], [Washington], [Utah], [Nevada],  
 [South], [Texas], [Oklahoma], [Louisiana], [New Mexico],  
 [Central], [Illinois], [Ohio], [Wisconsin], [Missouri], [Iowa], [Colorado] }
```

```
Descendants([Market], -1, SELF_BEFORE_AFTER)
```

prints a warning in application log, because a negative distance argument is not valid. The expression returns an empty set:

```
{}
```

```
Descendants([Market], 10, SELF)
```

returns an empty set, because there are no descendants of [Market] at a distance of 10 from [Market] level.

```
Descendants([Market], 10, BEFORE)
```

returns all descendants of [Market]:

```
{[Market],  
 [East], [New York], [Massachusetts], [Florida], [Connecticut], [New Hampshire]  
 [West], [California], [Oregon], [Washington], [Utah], [Nevada],  
 [South], [Texas], [Oklahoma], [Louisiana], [New Mexico],  
 [Central], [Illinois], [Ohio], [Wisconsin], [Missouri], [Iowa], [Colorado] }
```

```
Descendants([Market], 10, LEAVES)
```

returns all level-0 descendants of [Market]:

```
{[New York], [Massachusetts], [Florida], [Connecticut], [New Hampshire],  
 [California], [Oregon], [Washington], [Utah], [Nevada], [Texas],  
 [Oklahoma], [Louisiana], [New Mexico], [Illinois], [Ohio], [Wisconsin],  
 [Missouri], [Iowa], [Colorado]}
```

Distinct

Deletes duplicate tuples from a set.

Syntax

`Distinct (set)`

Parameter Description

`set` The set from which to remove duplicates.

Notes

- Duplicates are eliminated from the tail of the set.
- `Distinct` of an empty set returns an empty set.

Example

The expression

```
Distinct({[Colas], [Root Beer], [Cream Soda], [Colas]})
```

returns the set

```
{[Colas], [Root Beer], [Cream Soda]}
```

Note that the duplicate `[Colas]` is removed from the end of the set.

Dimension

Returns the dimension that contains the input element.

Syntax

```
member.Dimension
```

```
layer.Dimension
```

```
Dimension ( member | layer )
```

Parameter Description

`member` A member specification. The dimension returned is the dimension that this member belongs to.

`layer` A layer specification. The dimension returned is the dimension that this layer belongs to.

Example

`[Colas].Dimension` returns `Product`.

`[Market].[Region].Dimension` returns `Market`.

DrilldownByLayer

Drills down members of a set that are at a specified layer.

Syntax

```
DrilldownByLayer ( set [, layer | index ] )
```

Parameter Description

- set** The set in which the drilldown should occur.
- layer** The layer of the members that should be drilled down.
- index** A number of hierarchical steps representing the location of members that should be drilled down.

Notes

This function returns the members of *set* to one level below the optionally specified *layer* (or *index* number of the level). If *layer* (or *index*) is omitted, the lowest level of *set* is returned. Members are returned in their hierarchical order as represented in the database outline.

Example

The following query

```
SELECT
DrilldownByLayer (
  {[Product], [California]}, ([Product], [Oregon]),
  ([Product], [New York]), ([Product], [South]),
  ([Product], [Washington])}, [Market].[Region]
)
ON COLUMNS
FROM Sample.Basic
```

returns the grid:

Product								
California	Oregon	New York	South	Texas	Oklahoma	Louisiana	New Mexico	Washington
12964	5062	8202	13238	6425	3491	2992	330	4641

TO use *index*, note that *index* is the index number of the dimension to drill down on. In the example below, the function drills down on Market. If you change the 1 to a 0, it drills down on Product.

```
SELECT
DrilldownByLayer (
  {
    ([Product], [East]), ([Product], [West])
  }, 1
)
ON COLUMNS
FROM Sample.Basic
```

DrilldownMember

Drills down on any members or tuples of *set1* that are also found in *set2*. The resulting set contains the drilled-down members or tuples, as well as the original members or tuples (whether they were expanded or not).

Syntax

`DrilldownMember(set1, set2 [, RECURSIVE])`

Parameter	Description
-----------	-------------

<code>set1</code>	The set containing members or tuples to drill down on if comparison with <code>set2</code> tests positive for identical members or tuples.
-------------------	--------------------------------------------------------------------------------------------------------------------------------------------

<code>set2</code>	The set to compare with <code>set1</code> before drilling down on members or tuples in <code>set1</code> .
-------------------	------------------------------------------------------------------------------------------------------------

`RECURSIVE` Optional. A keyword to enable repeated comparisons of the sets.

Notes

This function drills down on all members of `set1` that are also found in `set2`. The two sets are compared. Then the members or tuples of the first set that are also present in the second set are expanded to include their children.

If the first set is a list of tuples, then any tuples in the first set that contain members from the second set are expanded to their children, generating more tuples.

If the `RECURSIVE` keyword is used, multiple passes are made on the expanded result sets. `Drilldownmember` repeats the set comparison and resulting drilldown until there are no more unexpanded members or tuples of `set1` that are also present in `set2`.

Example

Drilling Down on Members

The following examples drill down on members.

Example 1

Example 2

The following expression

```
DrilldownMember({Market, [New York]}, {Market, West}, RECURSIVE)
```

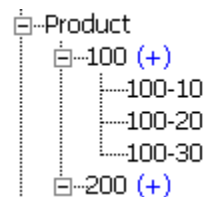
returns the set:

```
{Market, East, West, California, Oregon, Washington, Utah, Nevada, South,  
Central, [New York]}
```

The member `Market` is drilled down and then the `West` member of the resulting set is drilled down, because the `RECURSIVE` parameter was specified.

Drilling Down on Tuples

This example uses the following part of the Sample Basic outline:



The following example drills down on tuples.

The following expression

```
DrilldownMember
( {[100],[California]}, ([200],[Washington])),
  { [100] }
)
```

returns the set of tuples:

```
{ ([100],California), ([100-10],California), ([100-20],California),
  ([100-30],California), ([200],Washington)}
```

Therefore, the following query

```
SELECT
DrilldownMember
( {[100],[California]}, ([200],[Washington])),
  { [100] }
)
ON COLUMNS
FROM Sample.Basic
```

returns the grid:

100	100-10	100-20	100-30	200
California	California	California	California	Washington
999	3498	-1587	-912	1091

DrillupByLayer

Drills up the members of a set that are below a specified layer.

Syntax

```
DrillupByLayer ( set [,layer] )
```

Parameter Description

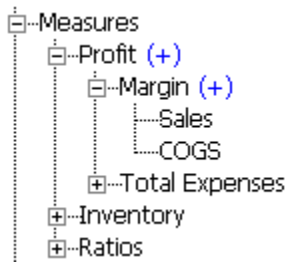
- set** The set in which the drill-up should occur.
- layer** The layer of the members that should be drilled up. If omitted, the set is drilled up to the second lowest level found in the set.

Notes

DrillupLevel can be used as a synonym for DrillupByLayer.

Example

These examples focus on the following hierarchy from the Sample Basic outline:



Example 1

The following query drills up the members of *set* to the second generation of the Measures dimension:

```

SELECT
  DrillupByLayer
  (
    {[Measures],[Profit],
     [Margin],[Sales],[COGS]}
  ), Generations([Measures], 2)
)

ON COLUMNS
FROM Sample.Basic

```

This query returns the grid:

Measures	Profit
105522	105522

Example 2

With no *layer* specified, the following query drills up the members of *set* to the second lowest level found in *set*:

```

SELECT
  DrillupByLayer
  (
    {[Measures],[Profit],
     [Margin],[Sales],[COGS]}
  )
)

ON COLUMNS
FROM Sample.Basic

```

This query returns the grid:

Measures	Profit	Margin
105522	105522	221519

DrillupMember

Tests two sets for common ancestors and drills up members of the first set to the level of the ancestors that are present in the second set.

Syntax

```
DrillupMember ( set1, set2 )
```

Parameter Description

- set1** The set containing members to drill up if comparison with *set2* tests positive for identical members or tuples.
- set2** The set to compare with *set1* before drilling up members in *set1*.

Notes

This function drills up any members of *set1* whose ancestors are found in *set2*. The level to which members in *set1* are drilled up depends on the level of the ancestor found in *set2*. The resulting set contains the ancestors of the drilled up member at the level found in *set2*, as well as any members of *set1* that were not drilled up.

Example

Example 1

The following example

```
DrillupMember({East, South, West, California, Washington, Oregon},{West})
```

returns the set:

```
{East, South, West}
```

The following expression

```
DrillupMember  
(  
  {East, South, West, California,  
    Washington, Oregon, Central, Nevada},  
  {West}  
)
```

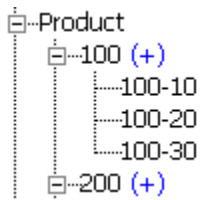
returns the set:

```
{East, South, West, Central, Nevada}
```

The member Nevada is not drilled up to member West because another member Central interrupts the chain of West descendants.

Example 2

The following examples use the following part of the Sample Basic outline:



The following expression

```
DrillupMember
({Product, [100], [100-10]},
 {Product})
)
```

returns the set:

```
{Product}
```

The following expression

```
DrillupMember
({Product, [100], [100-10]},
 {[100]})
)
```

returns the set:

```
{Product, [100]}
```

DTS

Calculates period-to-date values using built-in Dynamic Time Series functionality on block storage databases.

Syntax

DTS (dts-operation-specification, member)

Parameter	Description
dts-operation-specification	<p>The Dynamic Time Series member for which to return values. Specify one of the following operations:</p> <ul style="list-style-type: none"> ● HTD—History-to-date ● YTD—Year-to-date ● STD—Season-to-date ● PTD—Period-to-date ● QTD—Quarter-to-date ● MTD—Month-to-date ● WTD—Week-to-date ● DTD—Day-to-date

Note: The operation you use for this parameter must have a corresponding Dynamic Time Series member enabled in the outline.

Parameter	Description
<code>member</code>	Member specification. Must be a level-0 member from the time dimension.

Notes

This function is applicable only to block storage databases.

Example

The following query returns year to date information for Sample Basic.

```
WITH MEMBER [Year].[QuarterToDate_April] AS 'DTS(QTD, Apr) '
SELECT
  {[Profit], [Opening Inventory],[Ratios]}
ON COLUMNS,
  {[Jan], [Feb], [Mar], [Apr], [QuarterToDate_April]}
ON ROWS
FROM Sample.Basic;
```

This query returns the grid:

(axis)	Profit	Opening Inventory	Ratios
Jan	8024	117405	55.1017819772972
Feb	8346	116434	55.3868221647073
Mar	8333	115558	55.2665073107131
Apr	8644	119143	55.4181729805268
QuarterToDate_April	8644	119143	55.4181729805268

EnumText

Returns the text value corresponding to a numeric value in a text list.

Syntax

```
EnumText (textlistname, numeric_value_expression )
```

Parameter	Description
<code>textlistname</code>	Name of a text list defined on the outline.
<code>numeric_value_expression</code>	Numeric value expression (see “MDX Grammar Rules” on page 934).

Example

```
EnumText (CSRatings, 1)
```

returns “Excellent” if there is a text list named CSRatings containing the text “Excellent” mapped to ID 1. This example returns an empty string if there is no text associated with the given numeric ID.

EnumValue

Returns the internal numeric value for a text value in a text list.

Syntax

```
EnumValue (enum_string)
```

Parameter Description

enum_string Either *textlistname.string_literal* or *textlistmembername.string_literal*, where

- *textlistname* is the name of a text list defined on the outline
- *textlistmembername* is the name of a member that has an associated text list
- *string_literal* is the text value stored in the text list

Example

The following expression shows how EnumValue can be used to filter employees based on their title, which is stored as a text list in [Measures].[Title].

```
FILTER([Employee].Levels[0].Members, [Measures].[Title] = EnumValue([JobTitles]."Manager") )
```

Except

Returns a subset containing the differences between two sets, optionally retaining duplicates. The two input sets must have identical dimensionality.

Syntax

```
Except ( set1, set2 [,ALL] )
```

Parameter Description

set1 A set to compare with set2.

set2 A set to compare with set1.

ALL The optional ALL flag retains duplicates. Matching duplicates in set1 and set2 are eliminated.

Example

```
Except( {[New York], [California], [Florida], [California]},  
        {[Oregon], [Washington], [California], [Florida]})
```

returns {[New York]}.

```
Except( {[New York], [California], [Florida], [California]},  
        {[Oregon], [Washington], [California], [Florida]}, ALL)
```

returns {[New York], [California]}.

The following query returns Actual Sales and Profit numbers for the level-0 markets that are not defined as "Major Market."

```
SELECT
  {[Measures].[Sales], [Measures].[Profit]}
ON COLUMNS,
  Except (
    [Market].Levels(0).Members,
    UDA (Market, "Major Market")
  ) ON ROWS
FROM Sample.Basic
WHERE {[Year].[Qtr1], [Scenario].[Actual]}
```

This query returns the grid:

(axis)	Sales	Profit
Connecticut	3472	920
New Hampshire	1652	202
Oregon	5058	1277
Washington	4835	1212
Utah	4209	744
Nevada	6516	775
Oklahoma	2961	718
Louisiana	2906	773
New Mexico	1741	4
Wisconsin	4073	913
Missouri	3062	399
Iowa	6175	2036

Exp

Returns the exponent of an expression; that is, the value of e (the base of natural logarithms) raised to the power of the expression.

Syntax

```
Exp ( numeric_value_expression )
```

Parameter	Description
-----------	-------------

numeric_value_expression	A numeric value (see “MDX Grammar Rules” on page 934).
--------------------------	-------------------------------------------------------------------------

Notes

- Exp returns the inverse of Ln, the natural logarithm.
- The constant e is the base of the natural logarithm. e is approximately 2.71828182845904.

Example

The calculated member `Index` is created to represent e raised to the power of `[Variance %]/100`. In the example, `[Variance %]` divided by 100 is the numeric value expression provided to the `Exp` function.

```
WITH MEMBER [Scenario].[Index]
AS
    'Exp(
        [Scenario].[Variance %]/100
    )'
SELECT
    {[Scenario].[Variance %]}, [Scenario].[Index]}
ON COLUMNS,
    {[Market].children}
ON ROWS
FROM
    Sample.Basic
WHERE
    {[Sales]}
```

This query returns the grid:

(axis)	Variance %	Index
East	10.700	1.113
West	10.914	1.115
South	3.556	1.036
Central	3.595	1.037

See Also

- [Ln](#)

Extract

Returns a set of tuples with members from the specified dimensions of the input set.

Syntax

```
Extract ( set [, dimension ... ] )
```

Parameter Description

[set](#) The set from which to extract tuples belonging to the specified *dimension*.

[dimension](#) One or more dimensions from which to extract a set.

Notes

This function always removes duplicates. The *dimension* argument should specify dimensions present in the input set. It is an error to specify a dimension that is not present in the input set. The members in the tuples of the output set are ordered based on the dimension order specified in the input set.

Example

In the following example, Extract returns a subset of only those tuples belonging to the Year dimension.

```
SELECT
  Extract(
    {
      ([Year].[Qtr1], [Market].[California]),
      ([Year].[Qtr1], [Market].[Oregon]),
      ([Year].[Qtr2], [Market].[Oregon])
    }, Year
  )
ON COLUMNS
FROM Sample.basic
```

Qtr1	Qtr2
24703	27107

Factorial

Returns the factorial of a number.

Syntax

```
Factorial ( index )
```

Parameter Description

index A numeric value. The fractional part of *index* is ignored.

Example

Factorial(5) returns 120 (which is $5 * 4 * 3 * 2 * 1$).

Factorial(3.5) returns 6 (which is $3 * 2 * 1$). The fractional part of *index* is ignored.

Filter

Returns the tuples of a set that meet the criteria of a search condition.

Syntax

```
FILTER ( set, search_condition )
```

Parameter	Description
<code>set</code>	The set through which to iterate.
<code>search_condition</code>	A Boolean expression (see “ MDX Grammar Rules ” on page 934). The search condition is evaluated in the context of every tuple in the set.

Notes

This function returns the subset of tuples in *set* for which the value of the search condition is TRUE. The order of tuples in the returned set is the same as in the input set.

Example

Example 1

The following *unfiltered* query returns profit for all level-0 products:

```
SELECT
  { [Profit] }
ON COLUMNS,
  [Product].levels(0).members
ON ROWS
FROM Sample.Basic
```

This query returns the grid:

(axis)	Profit
100-10	22777
100-20	5708
100-30	1983
200-10	7201
200-20	12025
200-30	4636
200-40	4092
300-10	12195
300-20	2511
300-30	11093
400-10	11844
400-20	9851
400-30	-394
100-20	5708

(axis)	Profit
200-20	12025
300-30	11093

To filter the above results to only show negative Profit, use the Filter function, passing it the original set and a search condition. Filter will only return the set of members for which the search condition is true (for which Profit is less than zero).

```
SELECT
  { Profit }
ON COLUMNS,
  Filter( [Product].levels(0).members, Profit < 0)
ON ROWS
FROM Sample.Basic
```

The resulting query returns only the products with negative profit:

(axis)	Profit
400-30	-394

Example 2

The search expression in Example 1 compared a value expression (Profit) with a value. You can also filter using a member attribute as the search condition. For example, you can use the Filter function to only select members whose Caffeinated attribute is TRUE.

```
SELECT
  { [Profit] }
ON COLUMNS,
  Filter( [Product].levels(0).members, Product.CurrentMember.[Caffeinated])
ON ROWS
FROM Sample.Basic
```

This query returns profit for the members that are caffeinated:

(axis)	Profit
100-10	22777
100-20	5708
200-10	7201
200-20	12025
300-10	12195
300-20	2511
300-30	11093

To understand the search condition, `Product.CurrentMember.[Caffeinated]`, it may be helpful to read it right to left: Filter is searching for presense of the Caffeinated property on the

current member, for each member in the input set, which happens to be from the Product dimension (The CurrentMember function requires the dimension name as its argument).

Filter is an iterative function, meaning that at every member or tuple in the set being evaluated, the member being operated upon is the "current member," until Filter has looped through the entire input set and evaluated the search condition for each tuple. So to see how the previous query results were generated, it would be useful to see first which members actually have the Caffeinated attribute set to true. The following unfiltered query uses a calculated member to reveal which of the level-0 product members is caffeinated. The IIF function returns a value of 1 for each member whose Caffeinated attribute is set to TRUE, and returns a value of 0 otherwise.

```
WITH MEMBER Measures.IsCaffeinated
AS 'IIF(Product.CurrentMember.[Caffeinated], 1, 0)'
SELECT
    { IsCaffeinated }
ON COLUMNS,
    [Product].levels(0).members
ON ROWS
FROM Sample.Basic
```

This query returns the grid:

(axis)	IsCaffeinated
100-10	1
100-20	1
100-30	0
200-10	1
200-20	1
200-30	0
200-40	0
300-10	1
300-20	1
300-30	1
400-10	0
400-20	0
400-30	0
100-20	0
200-20	0
300-30	0

Looking at the results for the second query, you can begin to see that the search condition is evaluated for each tuple in the input set, and that only the tuples meeting the search condition are returned.

Example 3

Example 2 introduced the CurrentMember function. Even when CurrentMember is not explicitly called, Filter operates in the context of "the current member" while it iterates through a set. Filter and other iterative functions are processed in a nested context.

By default, Filter operates in the current-member context of top dimension members. You make the MDX context smaller by using a slicer (the Where clause), which overrides the built-in top-dimensional context. Additionally, you can override the slicer context by specifying context in the search condition argument for Filter.

The following query returns the Profit values for Western Region, for Qtr1. Note that the MDX context is West, Qtr1.

```
SELECT
  { [Profit] }
ON COLUMNS,
  [Product].levels(0).members
ON ROWS
FROM Sample.Basic
Where (West, Qtr1)
```

When adding a filter to the above query, the values for Profit are still evaluated as (Profit, West, Qtr1), because the sub-context for Filter is based on the main context.

```
SELECT
  { [Profit] }
ON COLUMNS,
  Filter( [Product].levels(0).members, Profit < 0 )
ON ROWS
FROM Sample.Basic
Where (West, Qtr1)
```

In the next query, the values for Profit are evaluated as (Profit, West, Qtr1), even though the outer context is (Profit, Market, Qtr1). This is because the inner context in the Filter function overrides the outer context of the slicer (West replaces Market).

```
SELECT
  { [Sales] }
ON COLUMNS,
  Filter( [Product].levels(0).members, (Profit, West) < 0 )
ON ROWS
FROM Sample.Basic
Where (Market, Qtr1)
```

The above query returns the Sales values for West, Qtr1 for members of Product whose Profit for West, Qtr1 was less than 0.

(axis)	Sales
100-20	2153

(axis)	Sales
400-30	1862
100-20	2153

Additional Examples

The following query on Sample Basic returns Qtr2 sales figures for products where the sales have increased by at least 10% since Qtr1.

```
SELECT
{
  Filter (
    [Product].Members,
    [Measures].[Sales] >
    1.1 *
    ( [Measures].[Sales], [Year].CurrentMember.PrevMember )
  )
}
on columns
FROM sample.basic
WHERE ([Year].[Qtr2], [Measures].[Sales])
```

Cola	Dark Cream
16048	11993

The following query on Sample Basic returns sales figures for product family "100" where the monthly sales of that product family are greater than 8,570. The filtering logic is stored as a named set in the WITH section.

```
WITH SET [High-Sales Months] as
,
  Filter(
    [Year].Levels(0).members,
    [Measures].[Sales] > 8570
  )
,
SELECT
  {[Measures].[Sales]}
ON COLUMNS,
  {[High-Sales Months]}
ON ROWS
FROM
  sample.basic
WHERE
  ([Product].[100])
```

(axis)	Sales
Apr	8685
May	8945

(axis)	Sales
Jun	9557
Jul	9913
Aug	9787
Sep	8844
Dec	8772

FirstChild

Returns the first child of the input member.

Syntax

```
member.FirstChild
```

```
FirstChild ( member )
```

Parameter Description

member A member specification. If a level-0 member, the output of FirstChild is an empty member.

Example

```
SELECT
  {[Qtr1].firstchild}
ON COLUMNS,
  {[Market].[Central].lastchild}
ON ROWS
FROM Sample.Basic
```

(axis)	Jan
Colorado	585

See Also

- [LastChild](#)
- [FirstSibling](#)

FirstSibling

Returns the first child of the input member's parent.

Syntax

```
FirstSibling ( member [, hierarchy ] )
```

```
member.FirstSibling [(hierarchy)]
```

Parameter Description

member A member specification.

hierarchy Optional. A specific hierarchy within the time dimension.

Notes

If *member* is the top member of a dimension, then *member* itself is returned.

Example**Example 1**

`Year.FirstSibling` returns `Year`.

`Qtr3.firstSibling` returns `Qtr1`.

Example 2

For every month, the following query displays the change in inventory level since the beginning of the quarter.

```
WITH MEMBER
  [Measures].[Inventory Level since beginning of Quarter]
AS
  '[Ending Inventory] - ([Opening Inventory], [Year].CurrentMember.FirstSibling) '
SELECT
  {[Measures].[Inventory Level since beginning of Quarter]}
ON COLUMNS,
  Year.Levels(0).Members ON ROWS
FROM Sample.Basic
```

This query returns the grid:

(axis)	Inventory Level Since Beginning of Quarter
Jan	-971
Feb	-1847
Mar	1738
Apr	6740
May	17002
Jun	24315
Jul	-871
Aug	-1243
Sep	-1608
Oct	2000

(axis)	Inventory Level Since Beginning of Quarter
Nov	5308
Dec	4474

See Also

- [LastSibling](#)
- [FirstChild](#)

FormatDate

Returns a formatted date-string.

Syntax

`FormatDate (date, internal-date-format)`

Parameter Description

date A number representing the input date between January 1, 1970 and Dec 31, 2037. The number is the number of seconds elapsed since midnight, January 1, 1970. To retrieve this number, use any of the following functions: `Today()`, `TodateEx()`, `GetFirstDate()`, `GetLastDate()`, `DateRoll()`.

Date-time attribute properties of a member can also be used to retrieve this number. For example,

- `Product.currentmember.[Intro Date]` returns the product introduction date for the current product in context.
- `[Cola].[Intro Date]` returns the product introduction date for Cola.

Parameter Description

internal-date-format	One of the following literal strings (excluding ordered-list numbers and parenthetical examples) indicating a supported date format. <ol style="list-style-type: none">1. "mon dd yyyy" (Example: mon = Aug)2. "Month dd yyyy" (Example: Month = August)3. "mm/dd/yy"4. "mm/dd/yyyy"5. "yy.mm.dd"6. "dd/mm/yy"7. "dd.mm.yy"8. "dd-mm-yy"9. "dd Month yy"10. "dd mon yy"11. "Month dd, yy"12. "mon dd, yy"13. "mm-dd-yy"14. "yy/mm/dd"15. "yymmdd"16. "dd Month yyyy"17. "dd mon yyyy"18. "yyyy-mm-dd"19. "yyyy/mm/dd"20. "Long format" (Example: "WeekDay, Mon dd, yyyy")21. "Short format" (Example: "m/d/yy")
----------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Notes

- Using an invalid input date returns an error.
- Using extra whitespace not included in the internal format strings returns an error.
- This function interprets years in the range 1970 to 2029 for yy format. Therefore, if the function is invoked using a date format mm/dd/yy for June 20, 2006, the returned date string is "06/20/06".

Example

The following query returns the first 10 day sales for all Colas products since their release date in the market California.

```
WITH MEMBER
Measures.[first 10 days sales] AS
'SUM(
    LastPeriods(-10,
        StrToMbr(
            FormatDate("Mon dd yyyy", Product.CurrentMember.[Intro Date])
        )
    )
)
```



```

    , Sales)'
SELECT
    {[first 10 days sales]}
ON COLUMNS,
    {Colas.Children}
ON ROWS
FROM MySamp.basic
WHERE (California, Actual);

```

Generate

Returns a set formed by evaluating a set expression. For each tuple in *set1*, return *set2*.

Syntax

```
Generate ( set1, set2 [, [ALL]] )
```

Parameter Description

- | | |
|-------------|------------------------------------------------------------------|
| <i>set1</i> | The set to loop through. |
| <i>set2</i> | The set expression to evaluate for every tuple in <i>set1</i> . |
| ALL | If the optional ALL flag is used, duplicate tuples are retained. |

Notes

The set expression *set2* is evaluated in the context of each of the tuples from *set1*. The resulting sets are combined, in the same order as of the tuples in *set1*, to produce the output. Duplicates are not included by default.

Example

For each region of the market, return its top-selling 3 products. Display the sales data by quarter.

```

WITH SET [Top3BevsPerRegion]
AS
    'Generate ({[Market].children},
    Crossjoin
    (
        {[Market].Currentmember},
        TopCount
        (
            [Product].Members, 3, [Measures].[Sales]
        )
    )
    )'
SELECT
    {[Top3BevsPerRegion]}
ON COLUMNS,
    {[Year].children}
ON ROWS
FROM Sample.Basic
WHERE ([Scenario].[Actual], [Measures].[Sales])

```

(axis)	East			West			South			Central		
(axis)	Product	Colas	Root Beer	Product	Diet Drinks	Cream Soda	Product	Root Beer	Diet Drinks	Product	Diet Drinks	Colas
Qtr1	20621	6292	5726	31674	8820	8043	12113	5354	4483	31412	10544	8074
Qtr2	224499	7230	5902	33572	9086	8982	12602	5535	4976	33056	10809	8701
Qtr3	22976	7770	5863	35130	9518	9616	13355	5690	4947	33754	10959	8894
Qtr4	21352	6448	6181	32555	8999	8750	12776	5429	4450	31458	10348	8139

Generation

Returns the generation of the input member.

Syntax

member.Generation

Parameter Description

member Member specification.

Example

The following query

```
SELECT
  [Year].[Qtr1].Generation.Members
ON COLUMNS,
  [Product].Generations(2).Members
ON ROWS
FROM Sample.Basic
```

returns the grid:

(axis)	Qtr1	Qtr2	Qtr3	Qtr4
100	7048	7872	8511	7037
200	6721	7030	7005	7198
300	5929	6769	6698	6403
400	5005	5436	5698	5162
Diet	7017	7336	7532	6941

See Also

- [Generations](#)
- [Level](#)
- [IsGeneration](#)

Generations

Returns the generation specified by the input generation number.

Syntax

```
dimension.Generations ( index )
```

```
Generations ( dimension, index )
```

Parameter Description

[dimension](#) The dimension specification.

`index` The numerical depth from the top member of the outline, where the top member is 1.

Example

The following query

```
SELECT
  [Year].[Qtr1].Generation.Members
ON COLUMNS,
  [Product].Generations(2).Members
ON ROWS
FROM Sample.Basic
```

returns the grid:

(axis)	Qtr1	Qtr2	Qtr3	Qtr4
100	7048	7872	8511	7037
200	6721	7030	7005	7198
300	5929	6769	6698	6403
400	5005	5436	5698	5162
Diet	7017	7336	7532	6941

See Also

- [Generation](#)
- [Levels](#)

GetFirstDate

Returns the start date for a date-time dimension member.

Syntax

```
GetFirstDate ( member )
```

Parameter Description

member A member from a date-time dimension.

Notes

- This function returns #MISSING if the input member is not from a date hierarchy in a Time-Date tagged dimension.
- The return value is a number representing the input date. The number is the number of seconds elapsed since midnight, January 1, 1970.
- This function is applicable only to aggregate storage databases.

Example

The following query returns sales for the first week of April, 2004.

```
SELECT
  {[Sales]}
ON COLUMNS,
  {DateToMember(
    GetFirstDate ([Apr 2004]),
    [Time dimension].Dimension,
    [Time dimension].[Weeks]
  )}
ON ROWS
FROM MySamp.basic;
```

GetFirstDay

For a given *date_part*, this function returns the first day of the time interval for the input date, following a standard Gregorian calendar.

Syntax

```
GetFirstDay ( date, date_part )
```

Parameter Description

date A number representing the input date between January 1, 1970 and Dec 31, 2037. The number is the number of seconds elapsed since midnight, January 1, 1970. To retrieve this number, use any of the following functions: Today(), TodateEx(), GetFirstDate(), GetLastDate(), DateRoll().

Date-Time type attribute properties of a member can also be used to retrieve this number. For example: Product.currentmember.[Intro Date] returns the Introduction or release date for the current product in context. [Cola].[Intro Date] returns the Introduction or release date for the “Cola” product.

date_part Defined time components of the standard calendar.

- DP_YEAR - year of the input date.
- DP_QUARTER – quarter of the input date.
- DP_MONTH - month of the input date.
- DP_WEEK - week of the input date.

Notes

This function can be used for getting the truncated date of an input date for a given date part, following a standard Gregorian calendar.

Example

Assuming today's date is April 15 2007, consider the following scenarios.

```
GetFirstDay( Today() , DP_YEAR)
```

returns the first day of the year, Jan 1 2007

```
GetFirstDay( Today() , DP_QUARTER)
```

returns the first day of the quarter, Apr 1 2007

```
GetFirstDay( Today() , DP_MONTH)
```

returns the first day of the month, Apr 1 2007

```
GetFirstDay( Today() , DP_WEEK)
```

returns the first day of the week, Apr 15 2007

See Also

- [GetNextDay](#)
- [GetLastDay](#)
- [Today](#)

GetLastDate

Returns the end date for a date-time dimension member.

Syntax

```
GetLastDate ( member )
```

Parameter Description

member A member from a date-time tagged dimension.

Notes

- This function returns #MISSING if the input member is not from a date hierarchy in a Time-Date tagged dimension.
- The return value is a number representing the input date. The number is the number of seconds elapsed since midnight, January 1, 1970.
- This function is applicable only to aggregate storage databases.

Example

The following query returns sales for the last week of April, 2004.

```
SELECT  
  {[Sales]}
```

```

ON COLUMNS,
  {DateToMember(
    GetLastDate ([Apr 2004]),
    [Time dimension].Dimension,
    [Time dimension].[Weeks]
  )}
ON ROWS
FROM MySamp.basic;

```

GetLastDay

For a given *date_part*, this function returns the last day of the time interval for the input date, following a standard Gregorian calendar.

Syntax

```
GetLastDay ( date, date_part )
```

Parameter Description

date A number representing the input date between January 1, 1970 and Dec 31, 2037. The number is the number of seconds elapsed since midnight, January 1, 1970. To retrieve this number, use any of the following functions: Today(), TodateEx(), GetFirstDate(), GetLastDate(), DateRoll().

Date-Time type attribute properties of a member can also be used to retrieve this number. For example: Product.currentmember.[Intro Date] returns the Introduction or release date for the current product in context. [Cola].[Intro Date] returns the Introduction or release date for the “Cola” product.

date_part Defined time components of the standard calendar.

- DP_YEAR - year of the input date.
- DP_QUARTER – quarter of the input date.
- DP_MONTH - month of the input date.
- DP_WEEK - week of the input date.

Notes

This function can be used for getting the truncated date of an input date for a given date part, following a standard Gregorian calendar.

Example

Assuming today’s date is April 15 2007, consider the following scenarios.

```
GetLastDay(Today(), DP_YEAR)
```

returns the last day of the year, Dec 31 2007

```
GetLastDay(Today(), DP_QUARTER)
```

returns the last day of the quarter, Jun 30 2007

```
GetLastDay(Today(), DP_MONTH)
```

returns the last day of the month, Apr 30 2007

```
GetLastDay(Today(), DP_WEEK)
```

returns the last day of the week, Apr 21 2007

See Also

- [GetFirstDay](#)
- [GetNextDay](#)
- [Today](#)

GetNextDay

To the given date and the week day, get the next date after input date that corresponds to the week day.

Syntax

```
GetNextDay ( date, week_day, [0|1] )
```

Parameter Description

date A number representing the input date between January 1, 1970 and Dec 31, 2037. The number is the number of seconds elapsed since midnight, January 1, 1970. To retrieve this number, use any of the following functions: [Today\(\)](#), [TodateEx\(\)](#), [GetFirstDate\(\)](#), [GetLastDate\(\)](#), [DateRoll\(\)](#).

Date-Time type attribute properties of a member can also be used to retrieve this number. For example: `Product.currentmember.[Intro Date]` returns the Introduction or release date for the current product in context. `[Cola].[Intro Date]` returns the Introduction or release date for the “Cola” product.

week_day A number between 1 (Sunday) and 7 (Saturday) representing the week day.

0 or 1 Optional. Indicates whether to include the date itself or not. Default behavior is 1: to include the date itself.

Example

```
GetNextDay(Today(), 2, 0)
```

returns the next Monday following today.

```
GetNextDay(Today(), 2, 1)
```

returns the next Monday following today, or today if today is Monday.

```
GetNextDay(Today(), 2)
```

returns the next Monday following today, or today if today is Monday.

See Also

- [GetFirstDay](#)
- [GetLastDay](#)
- [Today](#)

GetRoundDate

For a given *date_part*, this function returns the rounded date of the input date to the input time interval, following a standard Gregorian calendar.

Syntax

`GetRoundDate (date, date_part)`

Parameter Description

date A number representing the input date between January 1, 1970 and Dec 31, 2037. The number is the number of seconds elapsed since midnight, January 1, 1970. To retrieve this number, use any of the following functions: `Today()`, `TodateEx()`, `GetFirstDate()`, `GetLastDate()`, `DateRoll()`.

Date-Time type attribute properties of a member can also be used to retrieve this number. For example: `Product.currentmember.[Intro Date]` returns the Introduction or release date for the current product in context. `[Cola].[Intro Date]` returns the Introduction or release date for the “Cola” product.

date_part Defined time components of the standard calendar.

- `DP_YEAR` - year of the input date.
- `DP_QUARTER` – quarter of the input date.
- `DP_MONTH` - month of the input date.
- `DP_WEEK` - week of the input date.

Example

Assuming today’s date is April 15 2007, consider the following scenarios.

`GetRoundDate(Today(), DP_YEAR)`

returns the rounded date to the year, Jan 1 2007

`GetRoundDate(Today(), DP_QUARTER)`

returns the rounded date to the quarter, Apr 1 2007

`GetRoundDate(Today(), DP_MONTH)`

returns the rounded date to the month, Apr 1 2007

`GetRoundDate(Today(), DP_WEEK)`

returns the rounded date to the week, Apr 15 2007

See Also

- [GetNextDay](#)
- [GetFirstDay](#)
- [GetLastDay](#)
- [Today](#)

Head

Returns the first *n* members or tuples present in a set.

Syntax

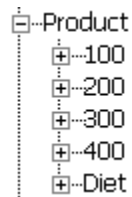
`Head (set [,numeric value expression])`

Parameter	Description
set	The set from which to take items.
numeric value expression	The count of items to take from the beginning of the set. If omitted, the default is 1. If less than 1, an empty set is returned. If the value exceeds the number of tuples in the input set, the original set is returned.

Example

Example 1

This example uses the following part of the Sample Basic outline:



The following expression

```
[Product].children
```

returns the set:

```
{ [100], [200], [300], [400], [Diet] }
```

Therefore, the following expression

```
Head (
  [Product].children, 2)

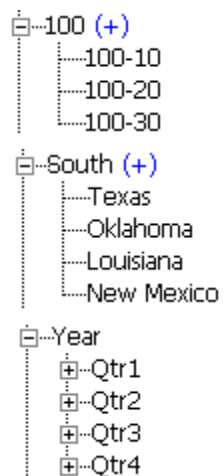
```

returns the first two members of the previous result set:

```
{ [100], [200] }
```

Example 2

This example uses the following parts of the Sample Basic outline:



The following expression

```
CrossJoin ( [100].children, [South].children )
```

returns the set:

```
{ ([100-10], Texas), ([100-10], Oklahoma), ([100-10], Louisiana), ([100-10], [New Mexico]),  
  ([100-20], Texas), ([100-20], Oklahoma), ([100-20], Louisiana), ([100-20], [New Mexico]),  
  ([100-30], Texas), ([100-30], Oklahoma), ([100-30], Louisiana), ([100-30], [New Mexico]) }
```

And the following expression

```
Head ( CrossJoin ([100].children, [South].children), 8 )
```

returns the first 8 tuples of the previous result set:

```
{ ([100-10], Texas), ([100-10], Oklahoma), ([100-10], Louisiana), ([100-10], [New Mexico]),  
  ([100-20], Texas), ([100-20], Oklahoma), ([100-20], Louisiana), ([100-20], [New Mexico]) }
```

Additionally, the following expression

```
([Year].generations(2).members)
```

returns the set of members comprising the second generation of the Year dimension:

```
{ [Qtr1], [Qtr2], [Qtr3], [Qtr4] }
```

Therefore, the following query

```
SELECT  
  {([Year].generations(2).members)}  
ON COLUMNS,  
Head (  
  CrossJoin (  
    [100].children, [South].children), 8  
  )  
ON ROWS  
FROM Sample.Basic
```

returns the grid:

(axis)		Qtr1	Qtr2	Qtr3	Qtr4
100-10	Texas	489	536	653	547
	Oklahoma	87	92	128	211
	Louisiana	93	106	128	137
	New Mexico	76	101	122	70

(axis)		Qtr1	Qtr2	Qtr3	Qtr4
100-20	Texas	206	199	152	82
	Oklahoma	84	66	55	79
	Louisiana	119	158	171	104
	New Mexico	-103	-60	-98	-18

See Also

- [Tail](#)

Hierarchize

Returns members of a set in their hierarchical order as represented in the database outline.

Syntax

```
Hierarchize ( set [,POST] )
```

Parameter Description

set Set specification.

POST If this keyword is used, child members are returned before their parents.

Notes

This function returns members of a set in their hierarchical order as represented in the database outline (viewed from top-down by default, meaning that parent members are returned before their children).

If POST is used, child members are returned before their parents (the view changes to bottom-up). For example,

```
Hierarchize({Child, Grandparent, Parent})
```

```
returns {Grandparent, Parent, Child}.
```

```
Hierarchize({Child, Grandparent, Parent}, POST)
```

```
returns {Child, Parent, Grandparent}.
```

Example

Example 1

The following expression

```
Hierarchize({May, Apr, Jun})
```

returns the set:

```
{Apr, May, Jun}
```

Therefore, the following query

```
Select  
Hierarchize({May, Apr, Jun})  
on columns from sample.basic
```

returns the grid:

Apr	May	Jun
8644	8929	9534

Example 2

The following expression

```
Hierarchize({May, Qtr2, Apr, Jun})
```

returns the set:

```
{ Qtr2 Apr May Jun }
```

Therefore, the following query

```
Select  
Hierarchize({May, Qtr2, Apr, Jun})  
on columns from sample.basic
```

returns the grid:

Qtr2	Apr	May	Jun
27107	8644	8929	9534

Example 3

The following expression

```
Hierarchize({May, Qtr2, Apr, Jun}, POST)
```

returns the set:

```
{Apr, May, Jun, Qtr2}
```

Therefore, the following query

```
Select  
Hierarchize({May, Qtr2, Apr, Jun}, POST)  
on columns from sample.basic
```

returns the grid:

Apr	May	Jun	Qtr2
8644	8929	9534	27107

Example 4

The following query

```

Select
Hierarchize({Dec, Year, Feb, Apr, Qtr1, Jun, Qtr2}, POST)
on columns,
Hierarchize({Margin, Sales})
on rows
from sample.basic

```

returns the grid:

(axis)	Feb	Qtr1	Apr	Jun	Qtr2	Dec	Year
Margin	17762	52943	18242	19457	56317	18435	221519
Sales	32069	95820	32917	35088	101679	33342	400855

IIF

Performs a conditional test, and returns an appropriate numeric expression or set depending on whether the test evaluates to true or false.

Syntax

```
IIF ( search_condition, true_part, false_part )
```

Parameter	Description
-----------	-------------

<i>search_condition</i>	An expression to evaluate as true or false (see “MDX Grammar Rules” on page 934).
-------------------------	----------------------------------------------------------------------------------------------------

<i>true_part</i>	A <i>value_expression</i> or a <i>set</i> . IIF returns this expression if the search condition evaluates to TRUE (something other than zero). The <i>value_expression</i> can be a numeric value expression or a string value expression.
------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<i>false_part</i>	A <i>value_expression</i> or a <i>set</i> . IIF returns this expression if the search condition evaluates to FALSE (zero). The <i>value_expression</i> can be a numeric value expression or a string value expression.
-------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Example

Example 1

The company plans an expensive promotion of its caffeinated drinks. For the Caffeinated products only, the following query calculates a Revised Budget that is 110% of the regular budget.

```

WITH MEMBER
  [Scenario].[Revised Budget]
AS
  'IIF (
    [Product].CurrentMember.Caffeinated,
    Budget * 1.1, Budget
  )'
SELECT
  {[Scenario].[Budget], [Scenario].[Revised Budget]}
ON COLUMNS,
  [Product].Levels(0).Members

```

```

ON ROWS
FROM Sample.Basic
WHERE ([Measures].[Sales], [Year].[Qtr3])

```

This query returns the grid:

(axis)	Budget	Revised Budget
100-10	18650	20515
100-20	8910	9801
100-30	3370	3370
200-10	11060	12166
200-20	9680	10648
200-30	3880	3880
200-40	2660	2660
300-10	10600	11660
300-20	3760	4136
300-30	8280	9108
400-10	7750	7750
400-20	6800	6800
400-30	3290	3290
100-20	8910	8910
200-20	9680	9680
300-30	8280	8280

Example 2

The following query calculates a Revised Budget equaling Budget for caffeinated products, and Actual for non-caffeinated products.

```

WITH MEMBER
  [Scenario].[Revised Budget]
AS
  'StrToMbr(IIF (
    [Product].CurrentMember.Caffeinated,
    "Budget" , "Actual"
  ))'
SELECT
  {[Scenario].[Budget], [Scenario].[Revised Budget]}
ON COLUMNS,
Children([100])
ON ROWS

```

```
FROM Sample.Basic
WHERE ([Measures].[Sales], [Year].[Qtr3])
```

This query returns the grid:

(axis)	Budget	Revised Budget
Cola	18650	18650
Diet Cola	8910	8910
Caffeine Free Cola	3370	3189

InStr

Returns a number specifying the position of the first occurrence of one string within another.

Syntax

```
InStr ( [start,] string1, string2 [,compare] )
```

Parameter Description

start	Optional character position to begin search in <i>string1</i> . The default value is 1. A position value of 1 indicates the very first character in the string. If omitted, search begins at first character in <i>string1</i> .
string1	String expression or literal string in which to search.
string2	String expression or literal string for which to search.
compare	Optional search mode. Values: 0 for case sensitive, 1 for case insensitive. Default is case sensitive.

Notes

If a matching string is not found, the return value is 0.

Example

```
InStr (5, "Year2000_promotional", "promotional", 1)
```

returns 10

Int

Returns the next lowest integer value of an expression.

Syntax

```
Int ( numeric_value_expression )
```

Parameter

Description

numeric_value_expression	A numeric value or an expression that returns a numeric value (see “MDX Grammar Rules” on page 934).
--------------------------	-----------------------------------------------------------------------------------------------------------------------

Example

Example 1

Int(104.504) returns 104.

Example 2

The following query

```
WITH MEMBER [Market].[West_approx]
AS
  'Int (
    Sum (
      Children([Market].[West])
    )
  ) '
SELECT
  {[Year].[Qtr1].Children}
ON COLUMNS,
  {[Market].[West].children,
  [Market].[West_approx]}
ON ROWS
FROM
  Sample.Basic
WHERE ([Measures].[Profit %], [Product].[Cola], [Scenario].[Actual])
```

returns the grid:

(axis)	Jan	Feb	Mar
California	38.643	37.984	38.370
Oregon	17.500	16.129	16.107
Washington	29.231	30.986	32.000
Utah	23.077	23.077	20.968
Nevada	-3.947	-6.757	-5.333
West_approx	104.000	101.00	102.000

Intersect

Returns the intersection of two input sets, optionally retaining duplicates.

Syntax

```
Intersect ( set1, set2 [,ALL] )
```

Parameter Description

set1 A set to intersect with *set2*.

Parameter Description

<code>set2</code>	A set to intersect with <i>set1</i> .
ALL	The optional ALL keyword retains matching duplicates in <i>set1</i> and <i>set2</i> .

Notes

Duplicates are eliminated by default from the tail of the set. The optional ALL keyword retains duplicates. The two input sets must have identical dimension signatures. For example, if *set1* consists of dimensions Product and Market, in that order, then *set2* should also consist of Product followed by Market.

Example

Example 1

The following expression

```
Intersect({[New York], [California], [Oregon]},
          {[California], [Washington], [Oregon]})
```

returns the set:

```
{[California], [Oregon]}
```

Therefore, the following query

```
SELECT
Intersect({[New York], [California], [Oregon]},
          {[California], [Washington], [Oregon]})
ON COLUMNS
FROM Sample.Basic
```

returns the grid:

California	Oregon
12964	5062

Example 2

The following expression

```
Intersect( { [New York], [California], [Florida], [California] },
           { [Oregon], [Washington], [California], [Florida], [California] }, ALL)
```

returns the set:

```
{ [California], [Florida], [California] }
```

Therefore, the following query

```

SELECT
Intersect( { [New York], [California], [Florida], [California] },
          { [Oregon], [Washington], [California], [Florida], [California] }, ALL)
ON COLUMNS
FROM Sample.Basic

```

returns the grid:

California	Florida	California
12964	5029	12964

The matching duplicate element [California] is duplicated in the result.

However, the following expression

```

Intersect( { [New York], [California], [Florida], [California] },
          { [Oregon], [Washington], [California], [Florida] }, ALL)

```

would return only

```
{ [California], [Florida] }
```

because only one match exists between [California] in set1 and [California] in set2.

Is

Returns TRUE if two members are identical.

Syntax

```
IS ( member1 , member2 )
```

```
member1 IS member2
```

Parameter Description

[member1](#) First member specification.

[member2](#) Second member specification.

Example

```
IS([Year].CurrentMember.Parent, [Qtr1])
```

returns TRUE if the parent of the current member in [Year] dimension is [Qtr1].

```
Filter([Year].Levels(0).members, IS([Year].CurrentMember.Parent, [Qtr1]))
```

returns children of [Qtr1].

The following query returns all members of [Market] that have the parent [East]; in other words, children of [East].

```

SELECT
{

```

```

Filter (
    [Market].members,
    [Market].CurrentMember.Parent IS [East]
)
}
on columns
FROM sample.basic

```

This query returns the following grid:

New York	Massachusetts	Florida	Connecticut	New Hampshire
8202	6712	5029	3093	1125

IsAccType

Returns TRUE if the member has the associated accounts tag. Account tags apply only to dimensions marked as Accounts dimensions. A FALSE value is returned for all other dimensions.

Syntax

```
IsAccType ( member , AcctTag )
```

Parameter Description

member A member specification.

AcctTag Valid values (defined in the database outline):

- First
- Last
- Average
- Expense
- TwoPass

Example

```

SELECT
Filter([Measures].Members, IsAccType([Measures].CurrentMember, First))
ON COLUMNS
FROM Sample.Basic

```

This query returns the following grid:

Opening Inventory
117405

IsAncestor

Returns TRUE if the first member is an ancestor of the second member and, optionally, if the first member is equal to the second member.

Syntax

IsAncestor (*member1* , *member2* [, INCLUDEMEMBER])

Parameter	Description
member1	A member specification.
member2	A member specification.
INCLUDEMEMBER	Optional. Use this keyword if you want IsAncestor to return TRUE if the first member is equal to the second member.

Example

Example 1

The following query returns all Market dimension members for which the expression `IsAncestor([Market].CurrentMember, [Florida])` returns TRUE; in other words, the query returns all ancestors of Florida.

```
SELECT
  Filter([Market].Members, IsAncestor([Market].CurrentMember, [Florida]))
ON COLUMNS
FROM Sample.Basic
```

Market	East
105522	24161

Example 2

The following query is the same as the above query, except that it uses INCLUDEMEMBER. It returns all Market dimension members for which the expression `IsAncestor([Market].CurrentMember, [Florida], INCLUDEMEMBER)` returns TRUE; in other words, the query returns Florida and all ancestors of Florida.

```
SELECT
  Filter([Market].Members, IsAncestor([Market].CurrentMember, [Florida], INCLUDEMEMBER))
ON COLUMNS
FROM Sample.Basic

{[Market], [East], [Florida]}
```

Market	East	Florida
105522	24161	5029

IsChild

Returns TRUE if the first member is a child of the second member and, optionally, if the first member is equal to the second member.

Syntax

IsChild (*member1* , *member2* [, INCLUDEMEMBER])

Parameter	Description
member1	A member specification.
member2	A member specification.

INCLUDEMEMBER Optional. Use this keyword if you want IsChild to return TRUE if the first member is equal to the second member.

Example

Example 1

The following query returns all Market dimension members for which the expression IsChild([Market].CurrentMember, [East]) returns TRUE; in other words, the query returns all children of East.

```
SELECT
  Filter([Market].Members, IsChild([Market].CurrentMember, [East]))
ON COLUMNS
FROM Sample.Basic
```

New York	Massachusetts	Florida	Connecticut	New Hampshire
8202	6712	5029	3093	1125

Example 2

The following query is the same as the above query, except that it uses INCLUDEMEMBER. It returns all Market dimension members for which the expression IsChild([Market].CurrentMember, [East]) returns TRUE; in other words, the query returns East and all children of East.

```
SELECT
  Filter([Market].Members, IsChild([Market].CurrentMember, [East], INCLUDEMEMBER))
ON COLUMNS
FROM Sample.Basic
```

East	New York	Massachusetts	Florida	Connecticut	New Hampshire
24161	8202	6712	5029	3093	1125

IsEmpty

Returns True if the value of an input numeric-value-expression evaluates to #MISSING, and returns FALSE otherwise.

Syntax

IsEmpty (*value_expression*)

Parameter Description

value_expression A set returning values to check for emptiness.

Notes

Zero is not equivalent to #MISSING. IsEmpty(0) returns TRUE.

Example

The following example finds all Product, Market combinations for which Sales data exists.

```
WITH SET [NewSet]
AS 'CrossJoin([Product].Children, [Market].Children)'
SELECT
  Filter([NewSet], NOT IsEmpty([NewSet].CurrentTuple))
ON COLUMNS
FROM Sample.Basic
WHERE
  {[Sales]}
```

This query returns the following grid:

100				...	400			Diet			
East	West	South	Central	...	East	West	Central	East	West	South	Central
27740	28306	16280	33808	...	15745	35034	33451	7919	36423	18676	42660

IsGeneration

Returns TRUE if the member is in a specified generation.

Syntax

```
IsGeneration ( member, index )
```

Parameter Description

member A member specification.

index A generation number.

Example

```
IsGeneration([Market].CurrentMember, 2)
```

returns TRUE if the current member of the Market dimension is at generation 2.

Therefore, the following query

```
SELECT
  Filter([Market].Members, IsGeneration([Market].CurrentMember, 2))
ON COLUMNS
FROM Sample.Basic
```

returns

East	West	South	Central
24161	29861	13238	38262

See Also

- [Generation](#)
- [IsLevel](#)

IsLeaf

Returns TRUE if the member is a level-0 member.

Syntax

IsLeaf (*member*)

Parameter Description

member A member specification.

Notes

IsLeaf(*member*) is the same as IsLevel(*member*, 0).

Example

```
IsLeaf([Market].CurrentMember)
```

returns TRUE if the current member of the Market dimension is at level 0.

Therefore, the following query

```
SELECT
  Filter([Market].Members, IsLeaf([Market].CurrentMember))
ON COLUMNS
FROM Sample.Basic
```

returns

New York	Massachusetts	Florida	...	Missouri	Iowa	Colorado
8202	6712	5029	...	1466	9061	7227

IsLevel

Returns TRUE if the member is in a specified level.

Syntax

IsLevel (*member* , *index*)

Parameter Description

`member` A member specification.
`index` A level number.

Example

```
IsLevel([Market].CurrentMember, 1)
```

returns TRUE if the current member of the Market dimension is at level 1.

Therefore, the following query

```
SELECT  
  Filter([Market].Members, IsLevel([Market].CurrentMember, 1))  
ON COLUMNS  
FROM Sample.Basic
```

returns

East	West	South	Central
24161	29861	13238	38262

See Also

- [Level](#)
- [IsGeneration](#)

IsMatch

Performs wild-card search / pattern matching to check if a string matches a given pattern. The input string can be a member name, an alias, an attribute value, or any relevant string. This function searches for strings matching the pattern you specify, and returns the artifacts it finds.

Syntax

```
IsMatch(string, patternstring, {MATCH_CASE | IGNORE_CASE})
```

Parameter Description

<code>string</code>	The string that should be tested against the pattern.
<code>patternstring</code>	The pattern to search for. Must be in POSIX Extended Regular Expression Syntax. See the syntax specification at The Open Group . See the Notes in this topic for additional rules regarding special characters.
<code>MATCH_CASE</code>	Optional. Consider <i>patternstring</i> to be case sensitive. If <code>MATCH_CASE</code> / <code>IGNORE_CASE</code> are omitted, Essbase defaults to the case-sensitive setting of the outline properties. To define database member names as case-sensitive, use Outline Editor in Administration Services (see the <i>Oracle Essbase Administration Services Online Help</i>).

Parameter	Description
-----------	-------------

IGNORE_CASE	Optional. Do not consider <i>patternstring</i> to be case sensitive. If MATCH_CASE / IGNORE_CASE are omitted, Essbase defaults to the case-sensitive setting of the outline properties.
-------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Notes

- To search for a member name containing \$, you must precede it with three backslash (\) escape characters in the *patternstring*. For example, to search for member a\$bc in Market, you must use `IsMatch(Market.CurrentMember.MEMBER_NAME, "a\\\$bc")`.
- To search for a character at the end of a line, you must precede the POSIX end-of-line anchor, which is a dollar sign (\$), with one backslash (\) escape character in the *patternstring*. For example, to search for a member name that ends with a c in Market, you must use `IsMatch(Market.CurrentMember.MEMBER_NAME, "c\$")`.
- To search for any other special characters besides \$, you must precede them with two backslash (\) escape characters in the *patternstring*. For example, to search for member a?bc in Market, you must use `IsMatch(Market.CurrentMember.MEMBER_NAME, "a\\?bc")`.

Example

The following query searches for members whose names start with “new”:

```
SELECT
  Filter(Market.Levels(0).Members,
    IsMatch(Market.CurrentMember.MEMBER_NAME, "^new")
  )
ON COLUMNS
FROM Sample.Basic
```

The following query searches for members whose names start with at least an “n”:

```
SELECT
  Filter(Market.Levels(0).Members,
    ISMATCH(Market.CurrentMember.MEMBER_NAME, "^n+")
  )
ON COLUMNS
FROM Sample.Basic
```

The following query searches for members whose names contain an “*”:

```
SELECT
  Filter(Year.Members,
    ISMATCH(Year.CurrentMember.MEMBER_NAME, "\\*")
  )
ON COLUMNS
FROM Sample.Basic
```

The following query searches for members whose names contain zero or an “a”:

```
SELECT
  Filter(Year.Members,
```

```

    ISMATCH(Year.CurrentMember.MEMBER_NAME, "a?")
)
ON COLUMNS
FROM Sample.Basic

```

IsSibling

Returns TRUE if the first member is a sibling of the second member and, optionally, if the first member is equal to the second member.

Syntax

```
IsSibling( member1, member2 [, INCLUDEMEMBER])
```

Parameter	Description
member1	A member specification.
member2	A member specification.

INCLUDEMEMBER Optional. Use this keyword if you want IsSibling to return TRUE if the first member is equal to the second member.

Example

Example 1

The following query returns all Market dimension members for which the expression `IsSibling([Market].CurrentMember, [California])` returns TRUE; in other words, the query returns all states that are siblings of California.

```

SELECT
  Filter([Market].Members, IsSibling([Market].CurrentMember, [California]))
ON COLUMNS
FROM Sample.Basic

```

Oregon	Washington	Utah	Nevada
5062	4641	3155	4039

Example 2

The following query is the same as the above query, except that it uses **INCLUDEMEMBER**. It returns all Market dimension members for which the expression `IsSibling([Market].CurrentMember, [California])` returns TRUE; in other words, the query returns all states that are siblings of California, including California itself.

```

SELECT
  Filter([Market].Members, IsSibling([Market].CurrentMember, [California],
INCLUDEMEMBER))
ON COLUMNS
FROM Sample.Basic

```

California	Oregon	Washington	Utah	Nevada
12964	5062	4641	3155	4039

IsUda

Returns TRUE if the member has the associated UDA tag (user-defined attribute).

Syntax

```
IsUda ( member , string_value_expression )
```

Parameter

Description

member

A member specification.

string_value_expression A user-defined attribute (UDA) name string, defined in the database outline.

Example

```
IsUda ([Market].CurrentMember, "Major Market")
```

returns TRUE if the current member of the Market has the user-defined attribute "Major Market."

Therefore, the following query

```
SELECT
  Filter ([Market].Members, IsUda ([Market].CurrentMember, "Major Market"))
ON COLUMNS
FROM Sample.Basic
```

returns

East	New York	Massachusetts	Florida	California	Texas	Central	Illinois	Ohio	Colorado
24161	8202	6712	5029	12964	6425	38262	12577	4384	7227

IsValid

Returns TRUE if the specified element validates successfully.

Syntax

```
IsValid ( member | tuple | set | layer | property )
```

Parameter Description

member

A member specification.

tuple

A tuple specification.

set

A set specification.

Parameter Description

layer A layer specification.

property A property specification (see “MDX Grammar Rules” on page 934).

Example

Example 1

The following example shows how `IsValid` can be used to check whether a given property value is valid. It returns all Product dimension members that have an Ounces attribute value of 12.

```
SELECT
Filter([Product].members,
      IsValid([Product].CurrentMember.Ounces)
      AND
      [Product].CurrentMember.Ounces = 12)
ON COLUMNS
FROM Sample.Basic
```

The expression `IsValid([Product].currentmember.Ounces)` returns TRUE for only those members in the Product dimension that have a valid property value for [Ounces]. This eliminates ancestral members such as [Product] and [Colas] that do not have the [Ounces] property defined because they are not level-0 members of the Product dimension.

The second part of the AND condition in the filter selects only those members with a value of 12 for [Ounces].

This query returns the following grid:

100-10	100-20	200-10	200-30	300-30
22777	5708	7201	4636	11093

Example 2

```
IsValid([Jan].FirstChild)
```

returns FALSE, because [Jan] is a level-0 member, therefore it does not have any children.

Item

Extracts a member from a tuple.

Extracts a tuple from a set.

Syntax

Syntax that Returns a Member—one of the following:

```
tuple[.Item] ( index )
```

```
Item ( tuple, index )
```

Syntax that Returns a Tuple—one of the following:

set[.Item] (*index*)

Item (set, *index*)

Parameter Description

tuple The tuple from which to get a member.

index The usage depends upon whether you are returning a member or a tuple:

- Returning a member: Numeric position (starting from 0) of the member to extract from the tuple. A valid value for *index* is from 0 to 1 less than the size of the input tuple. A value of less than 0, or greater than or equal to size of the input tuple, results in an empty member.
- Returning a tuple: Numeric position (starting from 0) of the tuple to extract from the set. A valid value for *index* is from 0 to 1 less than the size of the input set. A value of less than 0, or greater than or equal to size of the input set, results in an empty tuple.

set The set from which to get a tuple.

Example

Example 1, Extracting a Member from a Tuple

```
SELECT
{([Qtr1], [Sales], [Cola], [Florida], [Actual] ).Item(3)}
ON COLUMNS
FROM Sample.Basic
```

returns:

Florida

5029

```
SELECT
{Item(( [Qtr1], [Sales], [Cola], [Florida], [Actual] ), 2)}
ON COLUMNS
FROM Sample.Basic
```

returns:

Cola

22777

Example 2, Extracting a Tuple from a Set

The following query

```
SELECT
{CrossJoin
(
[Market].CHILDREN,
[Product].CHILDREN
).ITEM(0)}
ON COLUMNS
FROM Sample.Basic
```

returns the first tuple in the set `CrossJoin([Market].CHILDREN, [Product].CHILDREN)`, which is `([East], [Colas])`:

The above query can also be written as:

```
SELECT
{CrossJoin
 (
  [Market].CHILDREN,
  [Product].CHILDREN
 ) (0)}
ON COLUMNS
FROM Sample.Basic
```

because the `ITEM` keyword is optional.

Example 3, Extracting Member from a Set

Consider the following crossjoined set of Market and Product members:

```
{
([East], [100]), ([East], [200]), ([East], [300]), ([East], [400]), ([East], [Diet]),
([West], [100]), ([West], [200]), ([West], [300]), ([West], [400]), ([West], [Diet]),
([South], [100]), ([South], [200]), ([South], [300]), ([South], [400]), ([South], [Diet]),
([Central], [100]), ([Central], [200]), ([Central], [300]), ([Central], [400]), ([Central],
[Diet])
}
```

The following example

```
CrossJoin([Market].CHILDREN, [Product].CHILDREN).item(0)
```

returns the first tuple of the crossjoined set, `([East], [100])`, and the following example

```
CrossJoin([Market].CHILDREN, [Product].CHILDREN).item(0).item(1)
```

returns `[100]`, the second member of the first tuple of the crossjoined set.

JulianDate

To the given UNIX date, get its Julian date.

Syntax

```
JulianDate ( date )
```

Parameter Description

date	A number representing the input date between January 1, 1970 and Dec 31, 2037. The number is the number of seconds elapsed since midnight, January 1, 1970. To retrieve this number, use any of the following functions: <code>Today()</code> , <code>TodateEx()</code> , <code>GetFirstDate()</code> , <code>GetLastDate()</code> , <code>DateRoll()</code> .
------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Date-Time type attribute properties of a member can also be used to retrieve this number. For example: `Product.currentmember.[Intro Date]` returns the Introduction or release date for the current product in context. `[Cola].[Intro Date]` returns the Introduction or release date for the “Cola” product.

Notes

- This function is useful in converting the UNIX date to Julian Date or the 1900 Date system recognized by Microsoft Excel.
- In the 1900 date system, the first day that is supported is January 1, 1900. When you enter a date, the date is converted into a serial number that represents the number of elapsed days since January 1, 1900. For example, if you enter July 5, 1998, Microsoft Excel converts the date to the serial number 35981. By default, Microsoft Excel for Windows uses the 1900 date system.

Return Value

This function returns *juliandate*, a number representing the Julian date. This number is a continuous count of days and fractions elapsed since noon Universal Time on January 1, 4713 BC in the proleptic Julian calendar.

Note: For Excel workbooks using 1900 date system, (JulianDate – 2415018.50) gets the sequential serial number as per 1900 date system.

Example

The following query returns the total monthly sales for all Colas along with their release dates as in 1900 Date system in market “California” for “March 2007.”

```
WITH MEMBER
  Measures.[Product Intro Date]
AS
  'JulianDate(Product.CurrentMember.[Intro Date]) - 2415018.50'
SELECT
  {Measures.[Product Intro Date], Measures.Sales}
ON COLUMNS,
  {Colas.Children}
ON ROWS
FROM Sample.Basic
WHERE
  (California, [March 2007], Actual);
```

See Also

- [UnixDate](#)

Lag

Using the order of members existing in a database outline, returns a member that is *n* steps behind a given member, along the same generation or level (as defined by *layertype*).

Syntax

```
member.Lag (index [,layertype] [, hierarchy] )
```

```
Lag ( member, index [, hierarchy] )
```

Parameter Description

member The starting member from which .LAG counts to a given number of previous members.

index A number n representing how many steps prior to <member> to count.

layertype GENERATION or LEVEL. Generation is the default.

hierarchy Optional. A specific hierarchy within the time dimension.

Notes

- If the member specified by the Lag function does not exist, the result is an empty member. For example, using Sample Basic, [Jun] .lag (12) returns an empty member.
- When multiple hierarchies are enabled, this function returns NULL when the source member is in one hierarchy and the result member belongs to a different hierarchy.

Example

The following expression:

```
[Jun] .lag (3)
```

returns the member that is 3 steps prior to Jun:

```
[Mar]
```

The following expression:

```
[Jun] .lag (-3)
```

returns the member that is 3 steps following Jun:

```
[Sep]
```

For every month, the following query displays the sales and average over the last three months.

```
WITH MEMBER
  [Measures].[Average Sales in Last 3 months]
AS
  'Avg(
    { [Year].CurrentMember,
      [Year].CurrentMember.Lag(1),
      [Year].CurrentMember.Lag(2)
    },
    [Measures].[Sales]
  )'
SELECT
  { [Measures].[Sales],
    [Measures].[Average Sales in Last 3 months]
  }
ON COLUMNS,
  [Year].Levels(0).Members
ON ROWS
FROM Sample.Basic
```

This query returns the grid:

(axis)	Sales	Average Sales in Last 3 Months
Jan	31538	31538
Feb	23069	31803.500
March	32213	31940
April	32917	32399.667
May	33674	32934.667
Jun	35088	33893
Jul	36134	34965.333
Aug	36008	35743.333
Sep	33073	35071.667
Oct	32828	33969.667
Nov	31971	32624
Dec	33342	32713.667

See Also

- [Lead](#)
- [PrevMember](#)

LastChild

Returns the last child of the input member.

Syntax

member.LastChild

LastChild (*member*)

Parameter Description

member A member specification.

Example

```
SELECT
  {[Qtr1].firstchild}
ON COLUMNS,
  {[Market].[Central].lastchild}
ON ROWS
FROM Sample.Basic
```

(axis)	Jan
Colorado	585

See Also

- [FirstChild](#)
- [LastSibling](#)

LastPeriods

Returns a set of members ending either at the specified member or at the current member in the time dimension.

Syntax

LastPeriods (*numeric value expression* [, *member* [, *hierarchy*]])

Parameter	Description
numeric value expression	The number of members to return (see “ MDX Grammar Rules ” on page 934). If negative, <i>member</i> is treated as the starting point.
member	Optional. A member expression.
hierarchy	Optional. A specific hierarchy within the time dimension.

Example

Lastperiods(3, Apr) returns the set {Feb, Mar, Apr}.

Lastperiods(-3, Apr) returns the set {Apr, May, Jun}.

Lastperiods(1, Apr) returns a set of one member: {Apr}.

Lastperiods(0, Apr) returns an empty set.

Lastperiods(5, Apr) returns the set {Jan, Feb, Mar, Apr}. Note that the output set has only four members.

The following query:

```
WITH MEMBER
  [Measures].[Rolling Sales] AS
  'Avg (
    LastPeriods
      (3, [Year].Currentmember
      ),
    [Measures].[Sales]
  )'
SELECT
  {[Measures].[Sales], [Measures].[Rolling Sales]}
ON COLUMNS,
  Descendants ([Year].[Qtr2])
ON ROWS
FROM Sample.Basic
WHERE [Product].[Root Beer]
```

returns the grid:

(axis)	Sales	Rolling Sales
Qtr2	27401	27014
Apr	8969	8960
May	9071	8997
Jun	9361	9133.667

See Also

- [PeriodsToDate](#)
- [OpeningPeriod](#)
- [ClosingPeriod](#)
- [ParallelPeriod](#)

LastSibling

Returns the last child of the input member's parent.

Syntax

```
LastSibling ( member [, hierarchy ] )
```

```
member.LastSibling [ (hierarchy) ]
```

Parameter Description

member A member specification.

hierarchy Optional. A specific hierarchy within the time dimension.

Notes

If *member* is the top member of a dimension, then *member* itself is returned.

Example

Year.LastSibling returns Year.

Qtr3.LastSibling returns Qtr4.

See Also

- [FirstSibling](#)
- [LastChild](#)

Lead

Using the order of members existing in a database outline, returns a member that is *n* steps past a given member, along the same generation or level (as defined by *layertype*).

Syntax

```
member.Lead (index [,layertype ] [, hierarchy ])
```

```
Lead ( member, index [, hierarchy ] )
```

Parameter Description

member The starting member from which .LEAD counts a given number of following members.

index A number *n* representing how many steps away from <member> to count.

layertype GENERATION or LEVEL.

hierarchy Optional. A specific hierarchy within the time dimension.

Notes

- If the member specified by the Lead function does not exist, the result is an empty member. For example, using Sample Basic, [Jan].lead (12) returns an empty member.
- When multiple hierarchies are enabled, this function returns NULL when the source member is in one hierarchy and the result member belongs to a different hierarchy.

Example

The following expression:

```
[Jan].lead (11)
```

returns the member that is 11 steps past Jan:

```
[Dec]
```

The following expression:

```
[Dec].lead (-11)
```

returns the member that is 11 steps prior to Dec:

```
[Jan]
```

For every month, the following query displays the marketing expenses and budgeted sales for the next month.

```
WITH MEMBER
  [Measures].[Expected Sales in Next month]
AS
  '([Measures].[Sales], [Year].CurrentMember.Lead(1))'
SELECT
  {
    ([Scenario].[Actual], [Measures].[Marketing]),
    ([Scenario].[Budget], [Measures].[Expected Sales in Next month])
  }
ON COLUMNS,
[Year].Levels(0).Members
ON ROWS
FROM Sample.Basic
```

This query returns the grid:

(axis)	Actual	Budget
(axis)	Marketing	Expected Sales in Next Month
Jan	5223	30000
Feb	5289	30200
Mar	5327	30830
Apr	5421	31510
May	5530	32900
Jun	5765	33870
Jul	5985	33820
Aug	6046	31000
Sep	5491	29110
Oct	5388	29540
Nov	5263	30820
Dec	5509	#Missing

See Also

- [Lag](#)
- [NextMember](#)

Leaves

Returns the set of level 0 (leaf) members that contribute to the value of the specified member.

The Leaves function compactly describes large sets of members or tuples while avoiding pre-expansion of the set before retrieval. Because large sets tend to be very sparse, only a few members contribute to the input member (have non #Missing values) and are returned. As a result, Leaves consumes less memory resources than the equivalent nonempty Descendants function call, allowing for better scalability, especially in concurrent user environments.

Members with #MISSING values are not included in the return set.

When *member* is on the primary hierarchy, the return set is the set of descendants at level 0 that are nonempty.

The set returned by Leaves is the set of nonempty descendants at level 0, with a few differences. For example, when *member* is from an alternate hierarchy, the return set contains all primary, stored, level 0 members whose values are aggregated into *member's* value. These contributing members may be either:

- Direct descendants of *member* along the alternate hierarchy

- Members that contribute value to a direct descendant of *member* by means of a shared member

In most cases, the Leaves function does not pre-expand the set prior to retrieval. Thus it requires less memory resources than the Descendants function, allowing for more scalability in dealing with large sets, especially in a high-concurrency user environment. Large sets tend to be very sparse; therefore, very few members are returned given the current point of view as defined by the MDX current member stack.

For example, a healthcare provider may have a database containing Doctor and Geography dimensions. While there may be hundreds of thousands, even millions, of doctors, only a fraction have data associated with them for a given geographic location. Leaves is ideal for queries where the set is large but is sparse at a given point of view:

```
Select {[Copayments]} ON COLUMNS
CrossJoin(Leaves ([Doctors]), Leaves([Santa Clara County]) ON ROWS
```

The Leaves function is beneficial for queries on large dimensions.

In some cases, Leaves does require pre-expansion of sets, limiting the memory savings. Pre-expansion of sets likely will occur when the input member to Leaves is:

- On an Accounts dimension
- On a Time dimension
- On a dimension with fewer than 10,000 members

Syntax

```
Leaves ( member )
```

Parameter Description

member The member for which contributing leaf members are sought

Notes

- This function is applicable only to aggregate storage databases. Using Leaves() with a non aggregate-storage input member returns an error.
- Leaves() is supported only for members in stored hierarchies. Using Leaves with a member in a dynamic hierarchy returns an error.
- If you modify the return set of Leaves with a metadata function such as Head, Tail, or Subset, then the query is not optimized. For example, querying for half of the Leaves set reduces performance to about the same as for the nonempty Descendants function call.
- Leaves() is recommended for use on large, sparse dimensions. In general, use Leaves() to optimize performance when the input set contains 10,000 members or more. For smaller, denser input sets, using the NON EMPTY keyword on an axis with CrossJoin might improve performance.

Example

The following examples are based on the Asosamp.Sample database.

Example 1 (Leaves)

The following query returns the Units (items per package) for all level 0 Personal Electronics products for which the Units data is not #MISSING:

```
SELECT
{Units} ON COLUMNS,
Leaves([Personal Electronics]) ON ROWS
FROM [Asosamp.Sample]
```

Because Leaves returns nonempty, level 0 descendants, the above query is identical to the following query:

```
SELECT
{Units} ON COLUMNS,
NON EMPTY Descendants([Personal Electronics], [Products].Levels(0), SELF) ON ROWS
FROM [Asosamp.Sample]
```

These queries return the following grid:

(axis)	Items Per Package
Digital Cameras	3041
Camcorders	3830
Photo Printers	6002
Memory	23599
Other Accessories	117230
Boomboxes	10380
Radios	20009

[Handhelds] was omitted from the result set because it has a value of #MISSING for the measure Units.

Example 2 (Leaves)

For this example, a third hierarchy called [Small Items] was added to the Products dimension.



The following query

```
SELECT
{Units} ON COLUMNS,
Leaves ([Small Items]) ON ROWS
FROM [Asosamp.Sample]
```

Returns the the following grid:

(axis)	Items Per Package
Digital Cameras	3041
Camcorders	3830
Memory	23599
Other Accessories	117230

In addition to the primary members [Digital Cameras] and [Camcorders], Leaves also returned the primary members [Memory] and [Other Accessories], because these level-0 members contributed to [Small Items] via [Handhelds/PDAs].

Left

Returns a specified number (*length*) of characters from the left side of the string .

Syntax

```
Left ( string , length )
```

Parameter Description

string Input string.

length The number of characters to return from the left side of the input string.

Example

```
Left ("Northwind", 5)
```


returns North.

Len

Returns length of a string in terms of number of characters.

Syntax

```
Len ( string )
```

Parameter Description

string A string.

Example

Level

Returns the level of the input member.

Syntax

```
member.Level
```

Parameter Description

member A member specification.

Example

The following query

```
SELECT
  [Year].[Qtr1].Level.Members
ON COLUMNS,
  [Product].Levels(0).Members
ON ROWS
FROM Sample.Basic
```

returns the grid:

(axis)	Qtr1	Qtr2	Qtr3	Qtr4
100-10	5096	5892	6583	5206
100-20	1359	1534	1528	1287
100-30	593	446	400	544
200-10	1697	1734	1883	1887
200-20	2963	3079	3149	2834
200-30	1153	1231	1159	1093

(axis)	Qtr1	Qtr2	Qtr3	Qtr4
200-40	908	986	814	1384
300-10	2544	3231	3355	3065
300-20	690	815	488	518
300-30	2695	2723	2855	2820
400-10	2838	2998	3201	2807
400-20	2283	2522	2642	2404
400-30	-116	-84	-145	-49
100-20	1359	1534	1528	1287
200-20	2963	3079	3149	2834
300-30	2695	2723	2855	2820

See Also

- [Generation](#)
- [Levels](#)
- [IsLevel](#)

Levels

Returns the level specified by the input level number.

Syntax

```
dimension.Levels ( index )
```

```
Levels ( dimension, index )
```

Parameter Description

dimension The dimension specification.

index The number of steps up from the lowest level-0 member of the dimension. The count begins with zero at leaf members.

Example

The following query

```
SELECT
  [Year].[Qtr1].Level.Members
ON COLUMNS,
  [Product].Levels(0).Members
ON ROWS
FROM Sample.Basic
```

returns the grid:

(axis)	Qtr1	Qtr2	Qtr3	Qtr4
100-10	5096	5892	6583	5206
100-20	1359	1534	1528	1287
100-30	593	446	400	544
200-10	1697	1734	1883	1887
200-20	2963	3079	3149	2834
...
300-30	2695	2723	2855	2820

See Also

- [Level](#)
- [Generations](#)

LinkMember

Returns a member's shared member along a given hierarchy.

This function can be used instead of passing hierarchy arguments to Parent, Ancestor, FirstSibling, and LastSibling functions. This function works well in conjunction with Is* functions such as IsAncestor, IsChild, IsSibling, IsLevel, IsGeneration, and IsLeaf.

Syntax

member.LinkMember(*hierarchy*)

LinkMember(*member*, *hierarchy*)

Parameter Description

member A member specification

hierarchy Optional. A specific hierarchy within the time dimension.

Notes

- This function is applicable only to aggregate storage databases.
- If the primary hierarchy is passed to this function, it returns the primary member.
- If there is no shared member along the given hierarchy, this function returns an empty member.
- If a calculated member is passed to this function, the calculated member itself is returned.

Example

The following examples are based on ASOSamp.Sample.

The following MDX returns the member [HDTV] along the [High End Merchandise] hierarchy. By default, the primary instance of [HDTV] is used.

```
LinkMember([HDTV], [High End Merchandise])
```

The following MDX also returns the member [HDTV] along the [High End Merchandise] hierarchy. In this example, the input member is on the input hierarchy.

```
LinkMember([High End Merchandise].[HDTV], [High End Merchandise])
```

The following MDX returns the member [HDTV] along the [All Merchandise] hierarchy.

```
LinkMember([All Merchandise].[HDTV], [All Merchandise])
```

The following MDX returns an empty member, because there is no instance of [Digital Cameras] along the [High End Merchandise] hierarchy. The empty member has a value of #MISSING.

```
LinkMember([Digital Cameras], [High End Merchandise])
```

The following MDX also returns an empty member.

```
LinkMember([All Merchandise], [High End Merchandise])
```

The following MDX also returns an empty member.

```
LinkMember([Products], [High End Merchandise])
```

The following MDX returns [High End Merchandise].

```
LinkMember([High End Merchandise], [High End Merchandise])
```

Ln

Returns the natural logarithm (base e) of an expression.

Syntax

```
Ln ( numeric_value_expression )
```

Parameter	Description
-----------	-------------

<code>numeric_value_expression</code>	A numeric value (see “MDX Grammar Rules” on page 934).
---------------------------------------	-------------------------------------------------------------------------

Notes

- Ln returns the inverse of Exp.
- The constant e is the base of the natural logarithm. e is approximately 2.71828182845904.

Example

```
WITH MEMBER [Measures].[Ln_Sales]
AS
  'Ln([Measures].[Sales])'
SELECT
  {[Year].levels(0).members}
ON COLUMNS,
  {[Measures].[Sales], [Measures].[Ln_Sales]}
ON ROWS
```

```
FROM
  Sample.Basic
WHERE
  ([Market].[East], [Product].[Cola])
```

returns the following grid:

(axis)	Jan	Feb	...	Nov	Dec
Sales	1812	1754	...	1708	1841
Ln_Sales	7.502	7.470	...	7.443	7.518

See Also

- [Log](#)
- [Log10](#)
- [Exp](#)

Log

Returns the logarithm of an expression to a specified base.

Syntax

```
Log ( numeric_value_expression [,base] )
```

Parameter	Description
numeric_value_expression	A numeric value or an expression that returns a numeric value (see “ MDX Grammar Rules ” on page 934).
base	Optional. A number representing the base to use for the logarithm. If less than zero, zero, or close to 1, the Log function returns #MISSING. If omitted, the Log function calculates the base-10 logarithm. Log (Sales, 10) is equivalent to Log(Sales), and is also equivalent to Log10(Sales).

Example

Log (9, 3) returns 2.

Log10

Returns the base-10 logarithm of an expression.

Syntax

```
Log10 ( numeric_value_expression )
```

Parameter	Description
numeric_value_expression	A numeric value or an expression that returns a numeric value (see “ MDX Grammar Rules ” on page 934).

Example

Log10(1000) returns 3.

Lower

Converts upper-case string to lower-case.

Syntax

```
Lower ( string )
```

Parameter Description

string Input string.

Example

```
Lower (STRING)
```

returns string

See Also

- [Upper](#)

LTrim

Trims all whitespace on the left side of the string.

Syntax

```
LTrim ( string )
```

Parameter Description

string Input string.

Example

```
LTrim("     STRING")
```

returns "STRING"

Max

Returns the maximum of values found in the tuples of a set.

Syntax

```
Max ( set [, numeric_value_expression ] )
```

Parameter	Description
<code>set</code>	The set to search for values.
<code>numeric_value_expression</code>	Optional numeric value expression (see “ MDX Grammar Rules ” on page 934).

Notes

The return value of Max is #MISSING if either of the following is true:

- The input set is empty.
- All tuple evaluations result in #MISSING values.

Example

```
WITH
MEMBER [Measures].[Max Qtr2 Sales] AS
    'Max (
        {[Year].[Qtr2]},
        [Measures].[Sales]
    ) '
SELECT
{ [Measures].[Max Qtr2 Sales] } on columns,
{ [Product].children } on rows
FROM Sample.Basic
```

(axis)	Max Qtr2 Sales
Colas	27187
Root Beer	27401
Cream Soda	25736
Fruit Soda	21355
Diet Drinks	26787

Median

Orders the set according to the numeric value expression, and then returns the value of the set's median tuple.

Syntax

```
Median ( set, numeric_value_expr )
```

Parameter	Description
<code>set</code>	The set from which to get a median tuple value.
<code>numeric_value_expr</code>	A numeric value or an expression that returns a numeric value.

Notes

This function is a special case of the Percentile function where $n = 50$.

Example

The following query returns the median price for radios paid in all states last year.

```
WITH MEMBER
  [Geography].[Median Mkt Price]
AS
  'Median ( [Geography].Levels(2).Members, [Measures].[Price Paid]) '
SELECT
  { [Geography].[Median Mkt Price]}
ON COLUMNS
FROM
  ASOSamp.Sample
WHERE ([Products].[Radios], [Years].[Prev Year] )
```

MemberRange

Using the order of members existing in a database outline, returns a range of members inclusive of and between two members in the same generation or level.

Syntax

```
MemberRange ( member1, member2 [, layertype] [, hierarchy] )
```

member1:*member2*

Parameter Description

member1 The beginning point of the member range.

member2 The endpoint of the member range.

layertype GENERATION or LEVEL. Available only with function-style MemberRange () syntax. If omitted or if operator-style *member* : *member* syntax is used, the range of members returned is inclusive of and between two specified members of the same **generation**. If MemberRange(*member*, *member*, LEVEL) is used, the range of members returned is inclusive of and between two specified members of the same **level**.

hierarchy Optional. A specific hierarchy within the time dimension.

Notes

- If the two input members are not from the same generation or level, the result is an empty set.
- If the two input members are not from the same dimension, an error is returned.
- The order of the output resembles the order of the input. See Example 2.
- If the hierarchy argument is passed, *member1* and *member2* should belong to the same hierarchy. Otherwise, an empty set is returned.
- When multiple hierarchies are enabled, this function returns NULL when the range begins in one hierarchy and terminates in another hierarchy.

Example

Example 1 (MemberRange)

The following set:

```
{ [Year].[Qtr1], [Year].[Qtr2], [Year].[Qtr3], [Year].[Qtr4] }
```

is returned by both of the following examples:

```
MemberRange ( [Year].[Qtr1], [Year].[Qtr4] )
```

```
( [Year].[Qtr1] : [Year].[Qtr4] )
```

Example 2 (MemberRange)

```
[Jan] : [Mar]
```

returns:

```
{ [Jan], [Feb], [Mar] }
```

```
[Mar] : [Jan]
```

returns:

```
{ [Mar], [Feb], [Jan] }
```

Example 3 (MemberRange)

The following query

```
SELECT
  { [Measures].[Sales], [Measures].[Profit] }
ON COLUMNS,
  MemberRange([Year].[Feb], [Year].[Nov])
ON ROWS
FROM Sample.Basic
```

returns the grid:

(axis)	Sales	Profit
Feb	32069	8346
Mar	32213	8333
Apr	32917	8644
May	33674	8929
Jun	35088	9534
Jul	36134	9878
Aug	36008	9545
Sep	33073	8489

(axis)	Sales	Profit
Oct	32828	8653
Nov	31971	8367

See Also

- [RelMemberRange](#)

Members

Returns all members of the specified dimension or layer.

Syntax

dimension.Members | Members (*dimension*)

layer.Members | Members (*layer*)

Parameter Description

[dimension](#) A dimension specification.

[layer](#) A layer specification.

Example

This example focuses on the following part of the Sample Basic outline:



The following expression:

```
{ ([Market].members) }
```

returns the following set, which includes all descendant members of the Market dimension:

```
{
  Market, [New York], Massachusetts, Florida, Connecticut,
  [New Hampshire], East, California, Oregon, Washington,
  Utah, Nevada, West, Texas, Oklahoma, Louisiana, [New Mexico],
  South, Illinois, Ohio, Wisconsin, Missouri, Iowa, Colorado, Central
}
```

The following expression:

```
{ ([Market].levels(1).members) }
```

returns the following set, which includes one level of descendant members of the Market dimension:

```
{East, West, South, Central}
```

The following query assumes that level 1 of the Market dimension has an alias of Region:

```
Select  
{( [Market].[Region].members ) }  
on columns  
from Sample.Basic
```

This query returns the following grid:

East	West	South	Central
24161	29861	13238	38262

Min

Returns the minimum of values found in the tuples of a set.

Syntax

```
Min ( set [,numeric_value_expression ] )
```

Parameter

Description

set

The set to search for values.

numeric_value_expression Optional numeric value expression (see [“MDX Grammar Rules” on page 934](#)).

Notes

The return value of Min is #MISSING if either of the following is true:

- The input set is empty.
- All tuple evaluations result in #MISSING values.

Example

For every quarter, the following query displays the minimum monthly sales value.

```
WITH MEMBER  
  [Measures].[Minimum Sales in Quarter]  
AS  
  'Min ([Year].CurrentMember.Children, [Measures].[Sales])'  
SELECT  
  {[Measures].[Minimum Sales in Quarter]}  
ON COLUMNS,  
  [Year].Children  
ON ROWS  
FROM Sample.Basic
```

This query returns the grid:

(axis)	Minimum Sales in Quarter
Qtr1	31538
Qtr2	32917
Qtr3	33073
Qtr4	31971

Mod

Returns the modulus (remainder value) of a division operation.

Syntax

`Mod (numeric_value_expr_1, numeric_value_expr_2)`

Parameter	Description
<code>numeric_value_expr_1</code>	The number for which to find the remainder. Must be a numeric value or an expression that returns a numeric value (see “MDX Grammar Rules” on page 934).
<code>numeric_value_expr_2</code>	The divisor. Must be a numeric value or an expression that returns a numeric value (see “MDX Grammar Rules” on page 934).

Notes

The Essbase implementation of the function Mod returns the following values, which may be different from other vendors' implementations:

$\text{Mod}(n,k) = -\text{Mod}(-n,k)$, where $n < 0$
 $\text{Mod}(n,k) = \text{Mod}(n,-k)$, where $k < 0$

Example

```
WITH MEMBER [Measures].[Factor] AS
  'Mod ([Measures].[Margin %],[Measures].[Profit %])'
SELECT
  {
    [Measures].[Margin %],
    [Measures].[Profit %],
    [Measures].[Factor]
  }
ON COLUMNS,
  {[Year].[Qtr1].Children}
ON ROWS
FROM sample.basic
```

returns:

(axis)	Margin %	Profit %	Factor
Jan	55.102	25.44	4.217

(axis)	Margin %	Profit %	Factor
Feb	55.387	26.025	3.337
Mar	55.267	25.868	3.530

NextMember

Using the order of members existing in a database outline, returns the next member along the same generation or level.

Syntax

```
member.NextMember [ ( layertype ) ]
```

```
NextMember ( member [, layertype ] )
```

Parameter Description

member The starting member from which .NEXTMEMBER counts one member forward.

layertype GENERATION or LEVEL. The default is Generation.

Notes

- If the next member is not found, this function returns an empty member. For example, using Sample Basic, these would return an empty member: `Qtr4.nextmember` and `Year.nextmember`.
- When multiple hierarchies are enabled, this function returns NULL when the source member is in one hierarchy and the result member belongs to a different hierarchy.

Example

Example 1

The following expression:

```
[Jun].nextmember
```

returns the member that is one step further than Jun:

```
[Jul]
```

Example 2

The following query

```
/*
For January, PrevMember doesn't exist
For December, NextMember doesn't exist
*/
```

```
WITH
```

```
MEMBER
```

```

[Measures].[Delta from Previous Month]
AS
' [Measures].[Sales] -
  ([Measures].[Sales], [Year].CurrentMember.PrevMember)
,

MEMBER [Measures].[Delta from Next Month]
AS
' [Measures].[Sales] -
  ([Measures].[Sales], [Year].CurrentMember.NextMember)
,

SELECT
{ [Measures].[Sales],
  [Measures].[Delta from Previous Month],
  [Measures].[Delta from Next Month]
}
ON COLUMNS,

[Year].Levels(0).Members
ON ROWS

FROM Sample.Basic
WHERE
(
[Scenario].[Actual],
[Market].[East],
[Product].[100]
)

```

returns the grid:

(axis)	Sales	Delta from Previous Month	Delta from Next Month
Jan	2105	2105	44
Feb	2061	-44	-65
Mar	2126	65	-132
Apr	2258	132	-89
May	2347	89	-278
Jun	2625	278	-110
Jul	2735	110	62
Aug	2673	-62	311
Sep	2362	-311	268
Oct	2094	-268	28
Nov	2066	-28	-222
Dec	2288	222	2288

See Also

- [PrevMember](#)
- [Lead](#)

NonEmptyCount

Returns the count of the number of tuples in a set that evaluate to non #Missing values.

Syntax

```
NonEmptyCount ( set [,numeric_value_expression ] )
```

Parameter

Description

[set](#)

The set in which to count tuples.

[numeric_value_expression](#) Optional expression (see “[MDX Grammar Rules](#)” on page 934).

Notes

Each tuple is evaluated and included in the count returned by this function. If the numeric value expression is specified, it is evaluated in the context of every tuple, and the count of non #Missing values is returned.

Example

The following query

```
With  
Member [Measures].[Number Of Markets]  
as 'NonEmptyCount (Market.Levels(0).Members, Sales)'
```

```
Select  
{[Measures].[Number Of Markets]} on Columns,  
{[100].Children, [200].Children} on Rows  
FROM Sample.Basic
```

Returns the grid:

(axis)	Number of Markets
100-10	20
100-20	16
100-30	8
200-10	20
200-20	17
200-30	9
200-40	3

NonEmptySubset

Given an input set, `NonEmptySubset` returns a subset of that input set in which all tuples evaluate to nonempty. An optional value expression may be specified for the nonempty check.

This function can help optimize queries that are based on a large set for which the set of nonempty combinations is known to be small. `NonEmptySubset` reduces the size of the set in the presence of a metric; for example, you might request the nonempty subset of descendants for specific Units.

`NonEmptySubset` is used to reduce the size of a set before a subsequent analytical retrieval.

Syntax

```
NonEmptySubset (set [, value_expression [, dimension...]])
```

Parameter	Description
set	The set to reduce
value_expression	A value expression--ideally, a stored member or a simple formula. For each tuple in <i>set</i> , if <i>value_expression</i> is nonempty, the tuple is returned as part of the subset. Otherwise, it is removed.
dimension	One or more (comma-separated) dimensions from which to return the non-empty subset

Notes

Value_expression, if used, should be a stored member or simple formula. If *value_expression* is a complex formula, the retrieval of the nonempty subset is not optimized.

Example

The following example gets the bottom 10 products in terms of Units (items per package), and then returns the CrossJoin of that set and the level 0 members (zip codes) of [Albany - NY].

```
WITH SET Bottom_10
AS '
  BottomCount (
    Leaves (Products) ,
    10,
    Units
  )
'
SELECT
  {Units}
ON COLUMNS,
  NonEmptySubset (CrossJoin (Bottom_10, Leaves ([Albany - NY])))
ON ROWS
FROM Asosamp.Sample
```

This query returns the following grid:

(axis)	Items Per Package
Digital Cameras,12201	4
Camcorders,12201	3

(axis)	Items Per Package
Photo Printers, 12201	2
Digital Recorders, 12201	2
Desktops,12201	3
Digital Cameras,12212	5
Camcorders,12212	2
Photo Printers, 12212	3
Flat Panel, 12212	1
HDTV,12212	1
Home Theater, 12212	1
Desktops, 12212	2
Notebooks,12212	1
Digital Cameras,12223	1
Camcorders,12223	1
Photo Printers,12223	4
HTDV,12223	1
Notebooks,12223	1
Camcorders,12229	4
HDTV,12229	1
Home Theater,12229	3
Desktops,12229	1
Digital Cameras,12249	2
Photo Printers,12249	3
Projection TVs,12249	1
HDTV,12249	2
Home Theater,12249	1
Digital Recorders,12249	1
Notebooks,12249	1
Camcorders,12257	2

(axis)	Items Per Package
Photo Printers,12257	4
Projection TVs,12257	2
HDTV,12257	1
Home Theater,12257	3
Digital Recorders,12257	1

NTile

Returns a division number of a tuple in a set. This function only applies to aggregate storage databases.

Syntax

`NTile (member_or_tuple, set, number_of_divisions, numeric_value_expr)`

Parameter	Description
<code>member_or_tuple</code>	A member or a tuple .
<code>set</code>	The set to order.
<code>number_of_divisions</code>	The number of divisions to use in ordering the set.
<code>numeric_value_expr</code>	A numeric value or an expression that returns a numeric value.

Notes

- This function is applicable only to aggregate storage databases.
- This function orders the set by a numeric value, divides it into n equal divisions, and returns the division number that the given tuple is in.

Example

```
WITH
MEMBER [Measures].[7tile] AS
    'Ntile
      ([Measures].[Price Paid],
       { [Products].Levels(0).Members },
       7,
       [Measures].[Price Paid]
      )'
SELECT
{ [Measures].[Price Paid], [Measures].[7tile] } on columns,
{ [Products].Levels(0).Members } on rows
FROM ASOSamp.Sample
```

NumToStr

Converts a double-precision floating-point value into a decimal string. The number is formatted according to locale-specific conventions.

Syntax

```
NumToStr ( numeric_value_expression )
```

Parameter

Description

numeric_value_expression Numeric value expression (see “MDX Grammar Rules” on page 934).

Example

```
NumToStr(1)
```

returns "1.00".

OpeningPeriod

Returns the first descendant of a layer, or the first child of the Time dimension.

Syntax

```
OpeningPeriod ( [ layer [, member ] ] )
```

Parameter Description

layer A layer specification. If omitted, the first descendant of *member* is used. If *member* is omitted, the first child of the Time dimension is assumed.

member Optional. A member specification. If omitted, the first child of the Time dimension is assumed (for example, Qtr1 in Sample Basic).

Notes

The return value of this function varies depending on the input.

1. When no arguments are specified, the input member is assumed to be the current member of the Time dimension, and Openingperiod returns the first child of that member. Do not use this function without arguments if there is no dimension tagged as Time.
2. When both *layer* and *member* arguments are given as input, Openingperiod returns the first descendant of the input member at the input layer. For example, `Openingperiod(Year.generations(3), Qtr3)` returns Jul. If the input *member* and *layer* are the same layer, the output is the input member. For example, `Openingperiod(Year.generations(3), Jul)` returns Jul.
3. When only the *layer* argument is specified, the input member is assumed to be the current member of the dimension used in the layer argument. Openingperiod returns the first descendant of that dimension, at the input layer. For example, `Openingperiod(Year.generations(3))` returns Oct.

See Also

- [ClosingPeriod](#)
- [LastPeriods](#)
- [ParallelPeriod](#)
- [PeriodsToDate](#)

Order

Sorts members of a set in order based on an expression.

Syntax

```
Order ( set, string_expr | numeric_value_expression [,BASC | BDESC] )
```

Parameter	Description
set	The set to sort.
string_expr	String sorting criteria.
numeric_value_expression	Numeric sorting criteria (see “MDX Grammar Rules” on page 934).
BASC	If this keyword is used, the returned set is arranged in ascending order. Ascending order is the default even if no keyword is used.
BDESC	If this keyword is used, the returned set is arranged in descending order.

Notes

This function ignores missing values.

Example

The following query displays budgeted Sales and Marketing in Qtr2, and the display of products is sorted based on ascending Actual Sales in Qtr1.

```
SELECT
  CrossJoin(
    {[Scenario].[Budget]},
    {[Measures].[Marketing], [Measures].[Sales]}
  )
ON COLUMNS,
  Order(
    [Product].Levels(0).Members,
    ([Year].[Qtr1], [Scenario].[Actual])
  )
ON ROWS
FROM Sample.Basic
WHERE ([Year].[Qtr2])
```

This query returns the grid:

(axis)	Budget	Budget
(axis)	Marketing	Sales
400-30	510	3240
100-30	450	3400
300-20	550	3800
200-40	310	2830
200-30	550	4060
100-20	1160	8800
100-20	1160	8800
200-10	2090	10330
400-20	880	6590
300-10	1450	10080
300-30	1080	7880
300-30	1080	7880
400-10	790	7410
200-20	1080	9590
200-20	1080	9590
100-10	1800	17230

Ordinal

Returns a generation number or level number.

Syntax

`Ordinal (layer)`

Parameter Description

layer A layer specification for which to determine the ordinal.

Example

The following example prints generation number and level number for each member in the Product dimension. The value of calculated member [ProdGen] is a generation number because the input argument to the Ordinal function is a generation. The value of calculated member [ProdLev] is a level number because the input argument to the Ordinal function is a level.

```

WITH
  MEMBER [Measures].[ProdGen] AS
    'Ordinal([Product].CurrentMember.Generation)'
  MEMBER [Measures].[ProdLev] AS
    'Ordinal([Product].CurrentMember.Level)'
SELECT
  {[ProdGen], [ProdLev]} ON COLUMNS,
  [Product].Members ON ROWS
FROM Sample.Basic

```

This query returns the following grid:

(axis)	ProdGen	ProdLev
Product	3	0
100	2	1
100-10	3	0
100-20	3	0
100-30	3	0
200	3	0
200-10	2	1
200-20	3	0
200-30	3	0
200-40	3	0
300	2	1
300-10	3	0
300-20	3	0
300-30	3	0
400	2	1
400-10	3	0
400-20	3	0
400-30	3	0
Diet	2	1
100-20	3	0
200-20	3	0
300-30	3	0

ParallelPeriod

Returns a member from a prior time period as the specified or default time member.

Syntax

```
ParallelPeriod ( [layer [, index [, member [, hierarchy ]]])
```

Parameter Description

layer	Optional layer specification. If omitted, the same layer is assumed.
index	Number of time periods to count back in the specified layer.
member	Optional member specification. If omitted, the default member is assumed (for more information, see Defaultmember).
hierarchy	Optional. A specific hierarchy within the time dimension.

Notes

If *layer*, *index*, and *member* are present, this function determines the member ANCESTOR1, which is computed as

```
Ancestor(member, layer)
```

The member ANCESTOR2 is then computed as

```
Lag(ANCESTOR1, index)
```

The return value of this function is then computed as

```
Cousin(member, ANCESTOR2)
```

If *layer* and *index* are present and *member* is absent, *member* is taken to be the current member along the dimension associated with *layer*. The returned value is determined as above.

If only *layer* is present, *index* is taken to be 1, and *member* is taken to be the current member along the dimension associated with *layer*. The returned value is determined as above.

If *layer*, *index*, and *member* are all absent, *member* is taken to be CurrentMember along TIME Dimension, *index* is taken to be 1, and *layer* is taken to be the generation of the parent of *member*. The returned value is determined as above.

See Also

- [LastPeriods](#)
- [PeriodsToDate](#)
- [ClosingPeriod](#)
- [OpeningPeriod](#)

Parent

Returns a member's parent.

Syntax

```
member.Parent [(hierarchy) ]  
Parent ( member [, hierarchy ] )
```

Parameter Description

member A member specification.

hierarchy Optional. A specific hierarchy within the time dimension.

Example

Example 1

```
SELECT  
  {Parent ([100-10])}  
ON COLUMNS  
FROM  
  sample.basic
```

returns the parent of 100-10:

100
30468

Example 2

The following query uses Filter to find the months in which Sales for [Product].[100] are higher than 8,570. The Parent function is used with Generate to create a set consisting of the parents (quarters) of the high-sales months.

```
WITH SET [High-Sales Months] as  
,  
  Filter(  
    [Year].Levels(0).members,  
    [Measures].[Sales] > 8570  
  )  
,  
SELECT  
  {[Measures].[Sales]}  
ON COLUMNS,  
  Generate([High-Sales Months], { Parent([Year].CurrentMember) })  
ON ROWS  
FROM  
  sample.basic  
WHERE  
  ([Product].[100])
```

This query returns the grid:

(axis)	Sales
Qtr2	27187

(axis)	Sales
Qtr3	28544
Qtr4	25355

Percentile

Orders the set according to the numeric value expression, and then returns the value of the tuple that is at the given percentile.

This function only applies to aggregate storage databases.

Syntax

```
Percentile ( set, numeric_value_expr, percentile )
```

Parameter	Description
set	The set from which to get a tuple value.
numeric_value_expr	A numeric value or an expression that returns a numeric value.
percentile	A percentile. Must be between 0 and 100.

Notes

- This function is applicable only to aggregate storage databases.
- The returned value is such that n percent of the of the set members are smaller than it.

Example

```
WITH MEMBER [Measures].[Perc] AS
  'Percentile(Products.Levels(0).Members, [Measures].[Price Paid], 10) '
SELECT {[Measures].[Price Paid], [Measures].[Perc] } ON COLUMNS,
{ Products.Levels(0).Members } ON ROWS
FROM AsoSamp.Sample
```

PeriodsToDate

Returns a set of single-member tuples from a specified layer up to a given member in that layer (or up to the default member), or, returns members up to the current member of the Time dimension.

Syntax

```
PeriodsToDate ( [layer [, member [, hierarchy ]]] )
```

Parameter Description

layer	The layer to use as a beginning point.
member	The member to use as an ending point.

Parameter Description

[hierarchy](#) Optional. A specific hierarchy within the time dimension.

Notes

- If *layer* and *member* are present, this function determines the ANCESTOR of *member*, computed as `Ancestor(member, layer)`.

Consider the subtree rooted at the ANCESTOR. This function returns the set of all members along the same generation between the first descendant of ANCESTOR at input member's generation and the input member (inclusive of both.)

The return value of this function is the set of single-member tuples constructed from the members in the subtree rooted at ANCESTOR which are in the same layer as *member* and which are at or before the position of *member* within its layer. The order of tuples in the returned set is the same as the order of the members included in the input layer.

- If *layer* is present and *member* is absent, *member* is considered to be CurrentMember of the dimension that *layer* is associated with.
- If *layer* and *member* are both absent, *member* is considered to be the current member of the Time dimension, and *layer* is assumed to be the generation of the member's parent. Hence the return value is a set containing the left siblings of *member* and *member* itself.
- Using `Periodstodate(layer, member)` has the same effect as using the following nested functions:

```
MemberRange(  
    OpeningPeriod(  
        member.GENERATION,  
        Ancestor(member, layer)  
    )  
    : member  
)
```

Example

`PeriodsToDate (Year.Generations(1), May)` returns the set:

```
{ Jan, Feb, Mar, Apr, May }
```

`PeriodsToDate (Year.Generations(2), May)` returns the set:

```
{ Apr, May }
```

`PeriodsToDate (Year.Generations(3), May)` returns the set:

```
{ May }
```

See Also

- [OpeningPeriod](#)
- [ClosingPeriod](#)
- [ParallelPeriod](#)
- [LastPeriods](#)

Power

Returns the result of raising a number to a given power.

Syntax

```
Power ( numeric_value_expression, power )
```

Parameter

Description

numeric_value_expression An expression that returns a value (see “[MDX Grammar Rules](#)” on page 934).

power The power to which the numeric value expression is raised.

Example

Power (9, 2.5) returns 243.

PrevMember

Using the order of members existing in a database outline, returns the previous member along the same generation or level.

Note: When multiple hierarchies are enabled, this function returns NULL when the source member is in one hierarchy and the result member belongs to a different hierarchy.

Syntax

```
member.PrevMember [ ( layertype ) ]
```

```
PrevMember ( member [, layertype ] )
```

Parameter Description

member The starting member from which PrevMember counts one member back.

layertype GENERATION or LEVEL. The default is Generation.

Example

Example 1

The following expression

```
[Jun].prevmember
```

returns the member that is 1 step prior to Jun:

```
[May]
```

Example 2

The following query

```

/*
For January, PrevMember doesn't exist
For December, NextMember doesn't exist
*/

WITH

MEMBER
  [Measures].[Delta from Previous Month]
AS
  ' [Measures].[Sales] -
    ([Measures].[Sales], [Year].CurrentMember.PrevMember)
  '

MEMBER [Measures].[Delta from Next Month]
AS
  ' [Measures].[Sales] -
    ([Measures].[Sales], [Year].CurrentMember.NextMember)
  '

SELECT
  { [Measures].[Sales],
    [Measures].[Delta from Previous Month],
    [Measures].[Delta from Next Month]
  }
ON COLUMNS,

  [Year].Levels(0).Members
ON ROWS

FROM Sample.Basic
WHERE
  (
    [Scenario].[Actual],
    [Market].[East],
    [Product].[100]
  )

```

Returns the grid:

(axis)	Sales	Delta from Previous Month	Delta from Next Month
Jan	2105	2105	44
Feb	2061	-44	-65
Mar	2126	65	-132
Apr	2258	132	-89
May	2347	89	-278
Jun	2625	278	-110
Jul	2735	110	62
Aug	2673	-62	311

(axis)	Sales	Delta from Previous Month	Delta from Next Month
Sep	2362	-311	268
Oct	2094	-268	28
Nov	2066	-28	-222
Dec	2288	222	2288

See Also

- [NextMember](#)
- [Lag](#)

Rank

Returns the numeric position of a tuple in a set.

Syntax

```
Rank ( member_or_tuple, set [, numeric_value_expr [, ORDINALRANK | DENSERANK | PERCENTRANK ]] )
```

Parameter	Description
member_or_tuple	The member or tuple to rank.
set	The set containing the tuple to rank. Should not have duplicate members.
numeric_value_expr	Optional. Numeric sorting criteria.
ORDINALRANK	Optional. Rank duplicates separately.
DENSERANK	Optional. Rank with no gaps in ordinals.
PERCENTRANK	Optional. Rank on a scale from 0 to 1.

Notes

This function is applicable only to aggregate storage databases.

If no numeric value expression is given, this function returns the 1-based position of the tuple in the set.

If a numeric value expression is given, this function sorts the set based on the numeric value and returns the 1-based position of the tuple in the sorted set.

If an optional rank flag is given, this function sorts the set based on the numeric value and returns the 1-based position of the tuple in the sorted set according to the instructions in the flag. The meanings of the flags are:

- [no flag]: Default behavior. Ties are given the same rank, and the next member is the count of members. Example:(1,1,1,4,5)

- **ORDINALRANK:** Ties are decided by Essbase. Duplicates are considered different entities.
Example: (1,2,3,4,5).
- **DENSERANK:** Ties are given the same rank, but there are no gaps in ordinals. Example:
(1,1,1,2,3)
- **PERCENTRANK:** Rank values are scaled by the cumulative sum up to this member.
Example: (.1, .15, .34, .78, 1.0). Values range from 0.0 to 1.0.

In the cases where this function sorts the set, it sorts tuples in descending order, and assigns ranks based on that order (highest value has a rank of 1).

Example

Example 1

```
WITH MEMBER [Measures].[Units_Rank] AS
  'Rank(Products.CurrentMember, Products.CurrentMember.Siblings)'
SELECT
  {Units, [Price Paid], [Units_Rank]}
ON COLUMNS,
  { Products.Members } ON ROWS
FROM ASOSamp.Sample;
```

Example 2

```
WITH MEMBER [Measures].[Units_Rank] AS
  'Rank( Products.CurrentMember, Products.CurrentMember.Siblings) '
SELECT {Units, [Measures].[Units_Rank]}
ON COLUMNS,
  Union(Children([Televisions]),
        Children([Radios]))
ON ROWS
FROM ASOSamp.Sample;
```

RealValue

Returns a value for the specified member or tuple without the inherited attribute dimension context.

Syntax

tuple[.RealValue]

member[.RealValue]

Parameter Description

tuple A tuple for which to return a real value

member A member for which to return a real value

Example

The following query sorts level-0 members of the Product dimension by the real value of Sales without the attribute dimension (Ounces_12) context, in descending order, and returns their sales for Ounces_12.

```
SELECT
  {[Sales]}
ON COLUMNS,
  Order ([Product].Levels(0).Members,
    [Sales].REALVALUE, BDESC)
ON ROWS
FROM Sample.Basic
WHERE ([OUNCES_12]) ;
```

RelMemberRange

Returns a set that is based on the relative position of the specified member in the database outline.

Note: When multiple hierarchies are enabled, this function returns NULL when the range begins in one hierarchy and terminates in another hierarchy.

Syntax

```
RelMemberRange ( member, prevcount, nextcount, [, layertype] [, hierarchy] )
```

Parameter Description

member An input member in the set you want to return.

prevcount The number of members in the same layer specified by *layertype* prior to *member* to include in the return set.

nextcount The number of members in the same layer specified by *layertype* following *member* to include in the return set.

layertype GENERATION or LEVEL. If omitted, the default is GENERATION. Defines whether the set to be returned is based the same generation or on the same level as *member*.

hierarchy Optional. A specific hierarchy within the time dimension.

Example

The following examples are based on ASOSamp.Sample.

Example 1

```
SELECT
  RelMemberRange ([PORTLAND - OR], 1, 2)
ON COLUMNS
FROM asosamp.sample
```

This query returns the set:

{[PHOENIX - OR],[PORTLAND - OR],[POWERS - OR],[PRAIRIE CITY - OR]}

Example 2

RelMemberRange(Apr, 5, 0)

returns the set {Jan, Feb, Mar, Apr}. Note that the output set has only four members.

RelMemberRange(Apr, 5, 10)

returns the set {Jan, Feb, Mar, Apr, May . . . ,Dec}. Note that the output set has only four previous members and seven next members of Apr.

See Also

- [LastPeriods](#)

Remainder

Returns the fractional part of the numeric value expression.

Syntax

Remainder (*numeric_value_expression*)

Parameter

Description

numeric_value_expression A numeric value expression (see “MDX Grammar Rules” on page 934).

Example

Remainder([Margin %])

extracts the fractional part of the [Margin %] value.

The following query shows [Margin %] and the fractional part of it for all members of the Product dimension.

```
WITH
  MEMBER [Measures].[Margin % Rem] AS 'Remainder([Margin %])'
SELECT
  {[Margin %], [Margin % Rem]} ON COLUMNS,
  [Product].Members ON ROWS
FROM Sample.Basic
```

This query returns the following grid:

(axis)	Margin %	Margin % Rem
Product	55.262	0.262
100	57.273	0.273
100-10	61.483	0.483
100-20	51.479	0.479

(axis)	Margin %	Margin % Rem
100-30	50.424	0.424
200	55.540	0.540
200-10	54.270	0.270
200-20	56.436	0.436
200-30	56.450	0.450
200-40	55.753	0.753
300	54.238	0.238
300-10	55.816	0.816
300-20	42.992	0.992
300-30	57.551	0.551
400	53.600	0.600
400-10	57.354	0.354
400-20	56.299	0.299
400-30	39.477	0.477
Diet	55.397	0.397
100-20	51.479	0.479
200-20	56.436	0.436
300-30	57.551	0.551

Right

Returns a specified number (*length*) of characters from the right side of the string .

Syntax

```
Right ( string , length )
```

Parameter Description

string Input string.

length The number of characters to return from the right side of the input string.

Example

```
Right ("Northwind", 4)
```

returns wind.

Round

Rounds a numeric value expression to the specified number of digits.

Syntax

```
Round ( numeric_value_expression, index )
```

Parameter	Description
<i>numeric_value_expression</i>	A numeric value expression (see “MDX Grammar Rules” on page 934).
<i>index</i>	Expression yielding an integer value. <i>numeric_value_expression</i> is rounded to the number of digits specified by this value. The fractional part of <i>index</i> is ignored.

Example

```
Round(234.5678, 2) returns 234.57.
```

RTrim

Trims all whitespace on the right side of the string.

Syntax

```
RTrim ( string )
```

Parameter	Description
<i>string</i>	Input string.

Example

```
RTrim("STRING   ")  
returns "STRING"
```

Siblings

Returns the siblings of the input member, optionally based on selection options.

Syntax

```
Siblings ( member[, selection [,include_or_exclude]] )  
member.Siblings
```

Parameter	Description
member	The member for which siblings are returned.

Parameter	Description
selection	<p>Optional. This option can be one of the following:</p> <ul style="list-style-type: none"> ● LEFT—Selects the siblings to the left of the input member ● RIGHT—Selects the siblings to the right of the input member ● ALL—Selects all the siblings of the input member <p>If no selection is made, the default is ALL.</p>
include_or_exclude	<p>Optional. This option can be one of the following:</p> <ul style="list-style-type: none"> ● INCLUDEMEMBER—Includes the input member in the siblings list ● EXCLUDEMEMBER—Excludes the input member from the siblings list <p>If neither is specified, the default is to include the input member.</p>

Notes

- If the input member is the top level of the dimension, this function returns a set containing the input member.
- In aggregate storage databases, in multiple-hierarchy-enabled dimensions, if the input member is a top-level member of a hierarchy, the output is members across hierarchies that are top-level members of hierarchies.
- This function is the same as `Children(member.parent)`.
- The `member.Siblings` syntax returns the same set as `Siblings(member)`, `Siblings(member, ALL)`, or `Siblings(member, ALL, INCLUDEMEMBER)`.

Example

Example 1

`Siblings(Year)` returns {Year}.

The following query

```
SELECT
CrossJoin (
    Union (
        Siblings ([Old Fashioned]),
        {[Root Beer]}, {[Cream Soda]}
    ),
    {(Budget)}, {[Variance]}
)
ON COLUMNS
from Sample.Basic
```

returns the grid:

Old Fashioned		Diet Root Beer		Sarsaparilla		Birch Beer		Root Beer		Cream Soda	
Budget	Variance	Budget	Variance	Budget	Variance	Budget	Variance	Budget	Variance	Budget	Variance
11640	-4439	14730	-2705	5050	-414	4530	-438	35950	-7996	29360	-3561

Example 2

The following examples are based on a Years – Quarters – Months Time hierarchy.

```
Siblings([Feb 2000], LEFT, INCLUDEMEMBER)
```

Returns {[Jan 2000], [Feb 2000]}.

```
Siblings([Feb 2000], RIGHT, EXCLUDEMEMBER)
```

Returns {[Mar 2000]}.

```
Siblings([Mar 2000], LEFT)
```

Returns {[Jan 2000], [Feb 2000], [Mar 2000]}.

```
Siblings([May 2000], RIGHT)
```

Returns {[May 2000], [Jun 2000]}.

```
Siblings([Mar 2000])
```

OR

```
[Mar 2000].Siblings
```

Returns {[Jan 2000], [Feb 2000], [Mar 2000]}.

Stddev

Calculates the standard deviation of the specified set. The calculation is based upon a sample of a population. Standard deviation is a measure of how widely values are dispersed from their mean (average).

Syntax

```
Stddev ( set [,numeric_value_expression [,IncludeEmpty] ] )
```

Parameter	Description
set	A valid MDX set specification.
numeric_value_expression	A numeric value or an expression that returns a numeric value (see “MDX Grammar Rules” on page 934).
IncludeEmpty	Use this keyword if you want to include in the calculation any tuples with #MISSING values. Otherwise, they are omitted by default.

Example

The following example, based on Sample Basic, calculates the standard deviation (based on a sample of a population) of the January sales values for all products sold in New York.

```
WITH MEMBER [Measures].[Std Deviation]
AS
    'Stddev(
        Crossjoin(
            {[Product].Children}, {[Measures].[Sales]}
        )
    )
'
,
SELECT
    {[Scenario].[Actual]}, [Scenario].[Budget]}
ON COLUMNS,
    {Crossjoin(
        {[Measures].[Sales]}, {[Product].Children}
    )},
    Crossjoin(
        {[Measures].[Sales]}, [Measures].[Std Deviation]},
        {[Product]}
    )}
ON ROWS
FROM
    Sample.Basic
WHERE
    ([Year].[Jan], [Market].[New York])
```

This query returns the following grid:

(axis)	Actual	Budget
(Sales, 100)	678	640
(Sales, 200)	551	530
(Sales, 300)	663	510
(Sales, 400)	587	620
(Sales, Diet)	#Missing	#Missing
(Sales, Product)	2479	2300
(Std Deviation, Product)	60.723	64.55

See Also

- [Stddevp](#)

Stddevp

Calculates the standard deviation of the specified set. This function assumes that the set represents the entire population. If you want to calculate based a sample of a population, use Stddev.

Standard deviation is a measure of how widely values are dispersed from their mean (average).

Syntax

```
Stddevp ( set [,numeric_value_expression [,IncludeEmpty] ])
```

Parameter	Description
set	A valid MDX set specification.
numeric_value_expression	A numeric value or an expression that returns a numeric value (see “MDX Grammar Rules” on page 934).
IncludeEmpty	Use this keyword if you want to include in the calculation any tuples with #MISSING values. Otherwise, they are omitted by default.

Example

The following example, based on Sample Basic, calculates the standard deviation (based on the entire population) of the January sales values for all products sold in New York.

```
WITH MEMBER [Measures].[Std Deviation]
AS
  'StddevP(
    Crossjoin(
      {[Product].Children}, {[Measures].[Sales]}
    )
  )
'
SELECT
  {[Scenario].[Actual]}, [Scenario].[Budget]}
ON COLUMNS,
  {Crossjoin(
    {[Measures].[Sales]}, {[Product].Children}
  ),
  Crossjoin(
    {[Measures].[Sales], [Measures].[Std Deviation]},
    {[Product]}
  )}
ON ROWS
FROM
  Sample.Basic
WHERE
  ([Year].[Jan], [Market].[New York])
```

This query returns the following grid:

(axis)	Actual	Budget
(Sales, 100)	678	640
(Sales, 200)	551	530
(Sales, 300)	663	510
(Sales, 400)	587	620

(axis)	Actual	Budget
(Sales, Diet)	#Missing	#Missing
(Sales, Product)	2479	2300
(Std Deviation, Product)	52.59	55.9

See Also

- [Stddev](#)

StrToMbr

Converts a string to a member name.

Syntax

```
StrToMbr ( string [, dimension ] [, MEMBER_NAMEONLY | alias_table_name ] )
```

Parameter	Description
string	Input string.
dimension	Optional dimension specification. If used, only member names found in this dimension will be returned.
MEMBER_NAMEONLY	Optional. Create member name only out of member names found (not including aliases). The default is to search for member names and all aliases.
alias_table_name	Optional. Create member name only out of alias name strings found. The default is to search for member names and all aliases.

Notes

You can also use member properties as string input. These properties include MEMBER_NAME, MEMBER_UNIQUE_NAME, MEMBER_ALIAS, ANCESTOR_NAMES, and COMMENTS.

For example:

```
SELECT {StrToMbr(Sales.MEMBER_NAME)} ON COLUMNS
FROM Sample.Basic
```

Example

```
SELECT
  { StrToMbr("CA" , [Geography], "Default") }
ON COLUMNS,
  Children([High End Merchandise])
ON ROWS
FROM Asosamp.Sample
```

returns CA.

```
SELECT
  { StrToMbr("Quarter1" , [Year], MEMBER_NAMEONLY) }
  DIMENSION PROPERTIES [YEAR].[MEMBER_ALIAS]
ON COLUMNS,
```

```
Children([100])
ON ROWS
FROM Sample.Basic
```

returns nothing, because "Quarter1" is an alias.

```
SELECT
  { StrToMbr("Qtr1" , [Year], MEMBER_NAMEONLY) }
  DIMENSION PROPERTIES [YEAR].[MEMBER_ALIAS]
ON COLUMNS,
  Children([100])
ON ROWS
FROM Sample.Basic
```

returns Qtr1.

```
SELECT
  { StrToMbr("Quarter1" , [Year], "Long Names") }
  DIMENSION PROPERTIES [YEAR].[MEMBER_ALIAS]
ON COLUMNS,
  Children([100])
ON ROWS
FROM Sample.Basic
```

returns Qtr1 because "Quarter1" is in the "Long Names" alias table.

StrToNum

Converts a string to a number.

Syntax

```
StrToNum (string)
```

Parameter Description

string Input string.

Notes

This function returns a numeric value after converting the string to a number. For example, string "0.9" becomes the number 0.9. StrToMbr returns zero if the string cannot be converted.

Example

```
StrToNum("0.9")
```

returns 0.9 as a numeric value expression.

Subset

Returns a subset from a set, in which the subset is a numerically specified range of tuples.

Syntax

```
Subset ( set, index1 [, index2 ] )
```


Parameter Description

<code>set</code>	The set from which to take tuples.
<code>index1</code>	The location of the tuple with which to begin the subset. Example: if <code>index1</code> is 0, the subset begins with the first tuple of <code>set</code> . If a negative value, the return is an empty set.
<code>index2</code>	Optional. The count of tuples to include in the subset. If omitted, all tuples to the end of <code>set</code> are returned. If a negative value, the return is an empty set. If the count goes beyond the range of the input set, all tuples to the end of the set are returned.

Notes

The first tuple of the subset is represented by `index1`. If `index1` is 0, then the first tuple of the returned subset will be the same as the first tuple of the input set.

Example

Example 1

The following expression

```
Subset ({Product.Members}, 0)
```

returns the set:

```
{ Product, [100-10], [100-20], [100-30], [100],  
  [200-10], [200-20], [200-30], [200-40], [200],  
  [300-10], [300-20], [300-30], [300],  
  [400-10], [400-20], [400-30], [400],  
  [100-20], [200-20], [300-30], Diet }
```

All tuples of the set `{Product.Members}` are returned, because the subset is told to begin with the first tuple, and no count of tuples given for `index2`.

Example 2

The following expression

```
Subset ({Product.Members}, 0, 4)
```

returns the set:

```
{ Product, [100], [100-10], [100-20] }
```

Therefore, the following query

```
Select  
  Subset ({Product.Members}, 0, 4)  
on columns  
from sample.basic
```

returns the grid:

Product	100	100-10	100-20
105522	30468	22777	5708

Substring

Returns the substring between a starting and ending position. Both the positional arguments are 1-based.

Syntax

```
Substring ( string, index1 [, index2 +] )
```

Parameter Description

string	String to subdivide (or field containing that string).
index1	A number <i>n</i> representing a starting position within a string.
index2	Optional. A number <i>n</i> representing an ending position within a string. If omitted, the endpoint is assumed to be the end of the original string.

Sum

Returns the sum of values of tuples in a set.

Syntax

```
Sum ( set [, numeric_value_expression ] )
```

Parameter

Description

set	The set containing the tuples to aggregate. If empty, the return value is #MISSING.
numeric_value_expression	Optional. An expression that returns a value. Commonly used to restrict the aggregation to a slice from a Measures dimension (see “ MDX Grammar Rules ” on page 934). In the example below, [Measures].[Total Expenses] is the numeric value expression provided to the Sum function.

Notes

For each tuple in *set*, the numeric value expression is evaluated in the context of that tuple and the resulting values are summed up.

The return value of Sum is #MISSING if either of the following is true:

- The input set is empty.
- All tuple evaluations result in #MISSING values.

Example

```
WITH MEMBER [Market].[Sum Expense for Main States]
AS
  'Sum
  ({[Market].[California], [Market].[Colorado],
  [Market].[Texas], [Market].[Illinois],
  [Market].[Ohio], [Market].[New York],
  [Market].[Massachusetts], [Market].[Florida]},
  [Measures].[Total Expenses]
  )'
```

```

SELECT
  {[Measures].[Total Expenses]}
ON COLUMNS,
  {UDA([Market], "Major Market"),
  [Market].[Sum Expense for Main States]}
ON ROWS
FROM
  Sample.Basic
WHERE ([Scenario].[Actual])

```

returns the grid:

(axis)	Total Expenses
New York	8914
Massachusetts	3412
Florida	5564
East	25310
California	11737
Texas	4041
Illinois	6900
Ohio	5175
Colorado	6131
Central	34864
Sum Expense for Main States	51874

Tail

Returns the last n members or tuples present in a set.

Syntax

```
Tail ( set [, index ] )
```

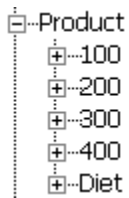
Parameter Description

<code>set</code>	The set from which to take items.
<code>index</code>	The number of items to take from the end of the set. If omitted, the default is 1. If less than 1, an empty set is returned. If the value exceeds the number of tuples in the input set, the original set is returned.

Example

Example 1

This example uses the following part of the Sample Basic outline:



The following expression

```
[Product].children
```

returns the set:

```
{ [100], [200], [300], [400], [Diet] }
```

Therefore, the following expression

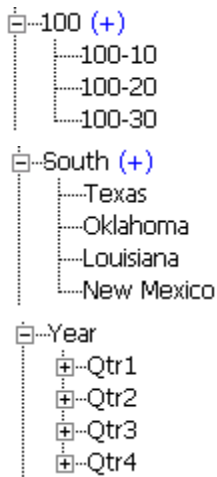
```
Tail (
  [Product].children, 2)
```

returns the last two members of the previous result set:

```
{ [400], [Diet] }
```

Example 2

This example uses the following parts of the Sample Basic outline:



The following expression

```
Crossjoin ( [100].children, [South].children )
```

returns the set:

```
{ ([100-10], Texas), ([100-10], Oklahoma), ([100-10], Louisiana), ([100-10], [New
Mexico]),
  ([100-20], Texas), ([100-20], Oklahoma), ([100-20], Louisiana), ([100-20], [New
Mexico]),
  ([100-30], Texas), ([100-30], Oklahoma), ([100-30], Louisiana), ([100-30], [New
Mexico]) }
```

And the following expression:

```
Tail ( Crossjoin ([100].children, [South].children), 8 )
```

returns the last 8 tuples of the previous result set:

```
{ ([100-20], Texas), ([100-20], Oklahoma), ([100-20], Louisiana), ([100-20], [New Mexico]),
  ([100-30], Texas), ([100-30], Oklahoma), ([100-30], Louisiana), ([100-30], [New Mexico]) }
```

Additionally, the following expression

```
([Year].generations(2).members)
```

returns the set of members comprising the second generation of the Year dimension:

```
{ [Qtr1], [Qtr2], [Qtr3], [Qtr4] }
```

Therefore, the following query

```
SELECT
  {[Year].generations(2).members}
ON COLUMNS,
  Tail (
    Crossjoin ([100].children, [South].children),
    8)
ON ROWS
FROM Sample.Basic
```

returns the grid:

(axis)	(axis)	Qtr1	Qtr2	Qtr3	Qtr4
100-20	Texas	206	199	152	82
	Oklahoma	84	66	55	79
	Louisiana	119	158	171	104
	New Mexico	-103	-60	-97	-18
100-30	Texas	#Missing	#Missing	#Missing	#Missing
	Oklahoma	#Missing	#Missing	#Missing	#Missing
	Louisiana	#Missing	#Missing	#Missing	#Missing
	New Mexico	#Missing	#Missing	#Missing	#Missing

To suppress the missing rows, use NON EMPTY at the beginning of the row axis specification:

```
SELECT
  {[Year].generations(2).members}
ON COLUMNS,
NON EMPTY
  Tail (
    Crossjoin ([100].children, [South].children),
    8)
ON ROWS
FROM Sample.Basic
```

This modified query returns as many of the 8 requested tuples as it can, without returning any that have entirely #Missing data:

(axis)		Qtr1	Qtr2	Qtr3	Qtr4
100-20	Texas	206	199	152	82
100-20	Oklahoma	84	66	55	79
100-20	Louisiana	119	158	171	104
100-20	New Mexico	-103	-60	-97	-18

See Also

- [Head](#)

Todate

Converts date strings to numbers that can be used in calculations.

Syntax

Todate (*string_value_expression_1* , *string_value_expression_2*)

Parameter	Description
string_value_expression_1	The format of the date string, either "mm-dd-yyyy" or "dd-mm-yyyy" (must be in lower case).
string_value_expression_2	The date string.

Notes

- If you specify a date that is earlier than 01-01-1970, this function returns an error.
- The latest date supported by this function is 12-31-2037.

Example

For products introduced before 06.01.1996, the following query calculates a Revised Budget that is 110% of Budget.

```
WITH MEMBER
  [Scenario].[Revised Budget]
AS
  'IIF (
    [Product].CurrentMember.[Intro Date]
    > TODATE("mm-dd-yyyy", "06-01-1996"),
    Budget * 1.1, Budget
  )'
SELECT
  {[Scenario].Budget, [Scenario].[Revised Budget]}
ON COLUMNS,
  [Product].[200].Children
  DIMENSION PROPERTIES [Intro Date]
ON ROWS
FROM Sample.Basic
WHERE ([Measures].[Sales], [Year].[Qtr3])
```

This query returns the grid:

Axis-1	Axis-1.properties	Budget	Revised Budget
200-10	(Intro Date = 09-27-1995, type: TIME,)	11060	11060
200-20	(Intro Date = 07-26-1996, type: TIME,)	9680	10648
200-30	(Intro Date = 12-10-1996, type: TIME,)	3880	4268
200-40	(Intro Date = 12-10-1996, type: TIME,)	2660	2926

TodateEx

Returns the numeric date value from input date-string according to the date-format specified. The date returned is the number of seconds elapsed since midnight, January 1, 1970.

If the date or the date format strings are invalid, an error is returned.

Syntax

`TodateEx (internal-date-format, date-string)`

Parameter Description

internal-date-format	<p>One of the following literal strings (excluding ordered-list numbers and parenthetical examples) indicating a supported date format.</p> <ol style="list-style-type: none">1. "mon dd yyyy" (Example: mon = Aug)2. "Month dd yyyy" (Example: Month = August)3. "mm/dd/yy"4. "mm/dd/yyyy"5. "yy.mm.dd"6. "dd/mm/yy"7. "dd.mm.yy"8. "dd-mm-yy"9. "dd Month yy"10. "dd mon yy"11. "Month dd, yy"12. "mon dd, yy"13. "mm-dd-yy"14. "yy/mm/dd"15. "yymmdd"16. "dd Month yyyy"17. "dd mon yyyy"18. "yyyy-mm-dd"19. "yyyy/mm/dd"20. Long format (Example: WeekDay, Mon dd, yyyy)21. Short format (Example: m/d/yy)
----------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Parameter Description

date-string A date string following the rules of *internal-date-format*. The following examples correspond to the above listed internal date formats.

1. Jan 15 2006
2. January 15 2006
3. 01/15/06
4. 01/15/2006
5. 06.01.06
6. 15/01/06
7. 15.01.06
8. 15-01-06
9. 15 January 06
10. 15 Jan 06
11. January 15 06
12. Jan 15 06
13. 01-15-06
14. 06/01/15
15. 060115
16. 15 January 2006
17. 15 Jan 2006
18. 2006-01-15
19. 2006/01/15
20. Sunday, January 15, 2006
21. 1/8/06 (m/d/yy)

Notes

- This function is an extension of [Todate](#).
- This function is case-sensitive. For example, using `apr` instead of `Apr` returns an error.
- Using extra whitespace not included in the internal format strings returns an error.
- Trailing characters after the date format has been satisfied are ignored. If you erroneously use a date string of `06/20/2006` with date format `mm/dd/yy`, the trailing `06` is ignored and the date is interpreted as June 20, 2020.
- Long Format (Weekday, Mon dd, yyyy) is not verified for a day-of-week match to the given date.

For example: For date string `Sunday, March 13, 2007` with date format `Long Format`, the input date string is parsed correctly for `March 13, 2007`, although `March 13, 2007` does not fall on Sunday.

- If you specify a date that is earlier than `01-01-1970`, this function returns an error.
- The latest date supported by this function is `12-31-2037`.

- When the `yy` format is used, this function interprets years in the range 1970 to 2029.

Example

The following query returns the actual sales on May 31, 2005 for the product Cola in the market California.

`TodateEx()` returns the date May 31, 2005, corresponding to date string `05.31.2005`. `StrToMbr` returns the corresponding day level member, capturing May 31, 2005.

```
SELECT
  {[Sales]}
ON COLUMNS,
  {
    StrToMbr(
      FormatDate(
        TodateEx("mm.dd.yyyy", "05.31.2005"),
        "Mon dd yyyy"
      )
    )
  }
ON ROWS
FROM Mysamp.basic
WHERE (Actual, California, Cola);
```

Today

Returns a number representing the current date on the Essbase computer. The number is the number of seconds elapsed since midnight, January 1, 1970.

Syntax

`Today`

Notes

The *date* returned can be used as input to other functions listed in the See Also section.

Example

This query returns today's actual sales for the product Cola in the market California. `Today()` returns today's date. `StrToMbr` retrieves the day member represented by the date returned by `Today`.

```
SELECT
  {[Sales]}
ON COLUMNS,
  {
    StrToMbr(
      FormatDate( Today(), "Mon dd yyyy")
    )
  }
ON ROWS
FROM Mysamp.basic;
```

See Also

- [DateToMember](#)
- [DateRoll](#)
- [DateDiff](#)
- [DatePart](#)
- [FormatDate](#)

TopCount

Returns a set of n elements ordered from largest to smallest, optionally based on an evaluation.

This function ignores missing values.

Syntax

```
TopCount ( set , index [,numeric_value_expression ] )
```

Parameter	Description
set	The set from which the top n elements are selected.
index	The number of elements to include in the set (n).
numeric_value_expression	Optional. An expression further defining the selection criteria (see “MDX Grammar Rules” on page 934).

Example

The following query selects the five top-selling markets in terms of yearly Diet products sales, and displays the quarterly sales for each Diet product.

```
SELECT
  CrossJoin(
    [Product].[Diet].Children,
    [Year].Children
  )
ON COLUMNS,
  TopCount(
    [Market].Levels(0).Members,
    5,
    [Product].[Diet]
  )
ON ROWS
FROM Sample.Basic
WHERE ([Scenario].[Actual], [Measures].[Sales])
```

This query returns the grid:

(axis)	100-20	100-20	100-20	100-20	200-20	200-20	200-20	200-20	300-30	300-30	300-30	300-30
(axis)	Qtr1	Qtr2	Qtr3	Qtr4	Qtr1	Qtr2	Qtr3	Qtr4	Qtr1	Qtr2	Qtr3	Qtr4
Illinois	755	958	1050	888	1391	1520	1562	1402	675	755	859	894

(axis)	100-20	100-20	100-20	100-20	200-20	200-20	200-20	200-20	300-30	300-30	300-30	300-30
California	367	491	506	468	1658	1833	1954	1706	700	802	880	673
Colorado	700	802	880	673	549	465	412	539	1006	921	892	991
Washington	637	712	837	704	459	498	597	514	944	799	708	927
Iowa	162	153	121	70	129	129	129	129	1658	1833	1954	1706

See Also

- [BottomCount](#)

TopPercent

Returns the smallest possible subset of a set for which the total results of a numeric evaluation are at least a given percentage. Elements in the result set are listed from largest to smallest.

Syntax

`TopPercent (set, percentage, numeric_value_expression)`

Parameter

Description

`set`

The set from which the top-percentile elements are selected.

`percentage`

The percentile. This argument must be a value between 0 and 100.

`numeric_value_expression`

The expression that defines the selection criteria (see “[MDX Grammar Rules](#)” on page 934).

Notes

This function ignores negative and missing values.

Example

The following query selects the top-selling markets that contribute 25% of the total yearly Diet products sales, and displays the quarterly sales for each Diet product.

```
SELECT
  CrossJoin(
    [Product].[Diet].Children,
    [Year].Children
  )
ON COLUMNS,
  TopPercent(
    [Market].Levels(0).Members,
    25,
    [Product].[Diet]
  )
ON ROWS
FROM Sample.Basic
WHERE ([Scenario].[Actual],
      [Measures].[Sales])
```

This query returns the grid:

(axis)	100-20	100-20	100-20	100-20	200-20	200-20	200-20	200-20	200-20	300-30	300-30	300-30	300-30
(axis)	Qtr1	Qtr2	Qtr3	Qtr4	Qtr1	Qtr2	Qtr3	Qtr4	Qtr1	Qtr2	Qtr3	Qtr4	
Illinois	755	958	1050	888	1391	1520	1562	1402	675	755	859	894	
California	367	491	506	468	1658	1833	1954	1706	700	802	880	673	
Colorado	700	802	880	673	549	465	412	539	1006	921	892	991	

TopSum

Returns the smallest possible subset of a set for which the total results of a numeric evaluation are at least a given sum. Elements of the result set are listed from largest to smallest.

Syntax

```
TopSum ( set, numeric_value_expression1, numeric_value_expression2 )
```

Parameter

Description

`set`

The set from which the highest-summing elements are selected.

`numeric_value_expression1` The given sum (see “MDX Grammar Rules” on page 934).

`numeric_value_expression2` The numeric evaluation (see “MDX Grammar Rules” on page 934).

Notes

- If the total results of the numeric evaluation do not add up to the given sum, an empty set is returned.
- This function ignores negative and missing values.

Example

The following query selects the top-selling markets that collectively contribute 60,000 to the total yearly Diet products sales, and displays the quarterly sales for each Diet product.

```
SELECT
  CrossJoin(
    [Product].[Diet].Children,
    [Year].Children
  )
ON COLUMNS,
  TopSum(
    [Market].Levels(0).Members,
    60000,
    [Product].[Diet]
  )
ON ROWS
FROM Sample.Basic
WHERE ([Scenario].[Actual],
  [Measures].[Sales])
```

This query returns the grid:

(axis)	100-20	100-20	100-20	100-20	200-20	200-20	200-20	200-20	300-30	300-30	300-30	300-30
(axis)	Qtr1	Qtr2	Qtr3	Qtr4	Qtr1	Qtr2	Qtr3	Qtr4	Qtr1	Qtr2	Qtr3	Qtr4
Illinois	755	958	1050	888	1391	1520	1562	1402	675	755	859	894
California	367	491	506	468	1658	1833	1954	1706	700	802	880	673
Colorado	700	802	880	673	549	465	412	539	1006	921	892	991
Washington	637	712	837	704	459	498	597	514	944	799	708	927
Iowa	162	153	121	70	129	129	129	129	1658	1833	1954	1706
Florida	620	822	843	783	548	611	657	577	332	323	260	159
Oregon	389	303	277	322	1006	921	892	991	263	231	197	184

Truncate

Returns the integral part of a number. The return value has the same sign as its argument.

Syntax

`Truncate (numeric_value_expression)`

Parameter	Description
-----------	-------------

<code>numeric_value_expression</code>	Numeric value expression (see “MDX Grammar Rules” on page 934).
---------------------------------------	----------------------------------------------------------------------------------

Example

`Truncate(2.65)` returns 2.

`Truncate(-8.12)` returns -8.

TupleRange

Returns the range of tuples between (and inclusive of) two tuples at the same level.

The range is created by identifying the level of the arguments and pruning the result set to include only the argument tuples and the tuples that are, in terms of outline order, between them.

Syntax

`TupleRange (tuple1, tuple2)`

Parameter	Description
-----------	-------------

<code>tuple1</code>	The first input tuple, marking the beginning of the range.
---------------------	------------------------------------------------------------

<code>tuple2</code>	The second input tuple, marking the end of the range.
---------------------	-------------------------------------------------------

Notes

- TupleRange serves the same purpose as the @XRANGE function in the Essbase calculator language.
- The two input tuples must be of the same dimensionality. See the example, wherein both input tuples are of the format ([Year],[Month]).

Example

TupleRange can be useful if you have two Time dimensions. For example, the following expression averages a value for the range of months from Mar 2005 to Feb 2006, inclusive.

```
AVG (
  TUPLERANGE(
    ([2005], [Mar]), ([2006], [Feb])
  )
)
```

The values are averaged for the following range:

```
{ ([2005], [Mar]),
  ([2005], [Apr]),
  ([2005], [May]),
  ([2005], [Jun]),
  ([2005], [Jul]),
  ([2005], [Aug]),
  ([2005], [Sep]),
  ([2005], [Oct]),
  ([2005], [Nov]),
  ([2005], [Dec]),
  ([2006], [Jan]),
  ([2006], [Feb]) }
```

Uda

Selects all members to which a specified user-defined attribute is associated in the entire dimension or in a subtree rooted at the input member.

Syntax

```
Uda ( dimension | member, string_value_expression )
```

Parameter	Description
dimension	The dimension in which matching UDAs are searched.
member	A member to search (descendants included) for matching UDAs.
string_value_expression	The name of the UDA to be selected. Can be an expression that evaluates to the UDA string, or an exact character string (not case-sensitive) enclosed in double quotation marks.

Notes

A user-defined attribute is a term associated with members of an outline to describe a characteristic. This function selects all members that have the specified UDA.

Example

Dimension Example

In the following query, the Uda function searches a dimension (top member included) for descendant members having a UDA of Major Market:

```
SELECT
  {[Measures].[Sales], [Measures].[Profit]} ON COLUMNS,
  {UDA([Market], "Major Market")} ON ROWS
FROM Sample.Basic
WHERE ([Year].[Jul], [Product].[Cola])
```

(axis)	Sales	Profit
East	2248	1156
New York	912	370
Massachusetts	665	564
Florida	286	104
California	912	370
Texas	567	206
Central	1392	369
Illinois	567	208
Ohio	85	18
Colorado	199	70

returning the grid:

Member Example

In the following query, the Uda function searches a member (itself included) for descendant members having a UDA of Major Market:

```
SELECT
  {[Measures].[Sales], [Measures].[Profit]} ON COLUMNS,
  {UDA([East], "Major Market")} ON ROWS
FROM Sample.Basic
WHERE ([Year].[Jul], [Product].[Cola])
```

returning the grid:

(axis)	Sales	Profit
East	2248	1156
New York	912	370
Massachusetts	665	564

(axis)	Sales	Profit
Florida	286	104

Union

Returns the union of two input sets, optionally retaining duplicates.

Syntax

```
Union ( set1, set2 [,ALL] )
```

Parameter Description

set1 A set to join with *set2*.

set2 A set to join with *set1*.

ALL If the optional ALL keyword is used, duplicates are retained.

Notes

Duplicates are eliminated by default from the tail of the set. The optional ALL keyword retains duplicates. The two input sets must have identical dimension signatures. For example, if *set1* consists of dimensions Product and Market, in that order, then *set2* should also consist of Product followed by Market.

Example

Example 1

The expression

```
Union( Siblings([Old Fashioned]), {[Sarsaparilla], [Birch Beer]})
```

returns the set

```
{ [Old Fashioned], [Diet Root Beer], [Sarsaparilla], [Birch Beer] }
```

Example 2

The expression

```
Union( Siblings([Old Fashioned]), {[Sarsaparilla], [Birch Beer]}, ALL)
```

returns the set

```
{ [Old Fashioned], [Diet Root Beer], [Sarsaparilla], [Birch Beer],  
  [Sarsaparilla], [Birch Beer] }
```

Example 3

The following query

```
SELECT  
CrossJoin (
```

```

    Union (
        Siblings ([Old Fashioned]),
        {[Root Beer]}, {[Cream Soda]}
    ),
    {(Budget), ([Variance])}
)

```

ON COLUMNS
from Sample.Basic

returns the grid

Old Fashioned		Diet Root Beer		Sarsaparilla		Birch Beer		Root Beer		Cream Soda	
Budget	Variance	Budget	Variance	Budget	Variance	Budget	Variance	Budget	Variance	Budget	Variance
11640	-4439	14730	-2705	5050	-414	4530	-438	35950	-7996	29360	-3561

UnixDate

To the given Julian date, get its UNIX date.

Syntax

```
UnixDate ( juliandate )
```

Parameter Description

juliandate A number representing the Julian date. This number is a continuous count of days and fractions elapsed since noon Universal Time on January 1, 4713 BC in the proleptic Julian calendar.

Note: For Excel workbooks using 1900 date system, (JulianDate – 2415018.50) gets the sequential serial number as per 1900 date system.

Notes

- This function is useful in converting the Julian date to UNIX date.
- In the 1900 date system, the first day that is supported is January 1, 1900. When you enter a date, the date is converted into a serial number that represents the number of elapsed days since January 1, 1900. For example, if you enter July 5, 1998, Microsoft Excel converts the date to the serial number 35981. By default, Microsoft Excel for Windows uses the 1900 date system.

Return Value

This function returns *date* a number representing the input date between January 1, 1970 and Dec 31, 2037. The number is the number of seconds elapsed since midnight, January 1, 1970. To retrieve this number, use any of the following functions: Today(), TodateEx(), GetFirstDate(), GetLastDate(), DateRoll().

Date-Time type attribute properties of a member can also be used to retrieve this number. For example: Product.currentmember.[Intro Date] returns the Introduction or release date for the current product in context. [Cola].[Intro Date] returns the Introduction or release date for the “Cola” product.

See Also

- [JulianDate](#)

Upper

Converts lower-case string to upper case.

Syntax

```
Upper ( string )
```

Parameter Description

string Input string.

Example

```
Upper(string)
```

returns STRING

See Also

- [Lower](#)

Value

Returns a value for the specified member or tuple.

Syntax

```
tuple[.Value]
```

```
member[.Value]
```

Parameter Description

tuple A tuple for which to return a value.

member A member for which to return a value.

Notes

The VALUE keyword is optional. In Example 2, the value of Sales can be represented either as [Sales].VALUE or [Sales]. Any value expression (for example, the value expressions supplied to functions such as Filter, Order, or Sum) has an implicit Value function in it. The expression [Qtr1] <= 0.00 is a shortcut for [Qtr1].VALUE <= 0.00.

Example

Example 1

```
[Sales].Value
```

Returns the value of the Sales measure.

```
([Product].CurrentMember, [Sales]).Value
```

Returns the value of the Sales measure for the current member of the Product dimension.

Note: The Value keyword is optional. The above expressions can also be entered as:

```
[Sales]
```

Which is equivalent to `[Sales].Value`

```
([Product].CurrentMember, [Sales])
```

Which is equivalent to `([Product].CurrentMember, [Sales]).VALUE`

Example 2

The following query sorts level-0 members of the Product dimension by the value of Sales, in descending order.

```
SELECT
  {[Sales]}
ON COLUMNS,
  Order([Product].Levels(0).Members,
    [Sales].VALUE, BDESC)
ON ROWS
FROM Sample.Basic
```

This query returns the grid:

(axis)	Sales
100-10	62824
300-10	46956
200-10	41537
200-20	38240
200-20	38240
300-30	36969
300-30	36969
400-10	35799
400-20	32670
100-20	30469
100-20	30469
200-30	17559

(axis)	Sales
300-20	17480
400-30	15761
100-30	12841
200-40	11750

WithAttr

Returns all base members that are associated with an attribute member of the specified type.

Syntax

`WithAttr (member, character_string_literal, value_expression)`

Parameter	Description
<code>member</code>	The top member of an attribute dimension.
<code>character_string_literal</code>	An operator. Must be enclosed in double quotation marks. The following operators are supported: <ul style="list-style-type: none"> ● > Greater than ● >= Greater than or equal to ● < Less than ● <= Less than or equal to ● = = Equal to ● <> or != Not equal to ● IN In
<code>value_expression</code>	An attribute value described by a value expression. The expression must evaluate to a numeric value for numeric/date attributes and must evaluate to a string for text valued attributes. Can also be an exact character string (not case-sensitive) enclosed in double quotation marks.

Example

The following query

```
SELECT
  Withattr([Pkg Type], "=", "Can")
on columns
FROM Sample.Basic
```

returns products that are packaged in a can:

Cola	Diet Cola	Diet Cream
22777	5708	11093

See Also

- [Attribute](#)

WithAttrEx

Returns the set of base members that are associated with a specified varying attribute member or dimension, given the perspective setting and the predicate.

Syntax

```
WithAttrEx ( member, options, character_string_literal, value_expression, ANY, tuple | member[, tuple | member] )
```

Parameter	Description
member	The top member of an attribute dimension.
<code>character_string_literal</code>	An operator. Must be enclosed in double quotation marks. The following operators are supported: <ul style="list-style-type: none">• > Greater than• >= Greater than or equal to• < Less than• <= Less than or equal to• = = Equal to• <> or != Not equal to• IN In
<code>value_expression</code>	An attribute value described by a value expression. The expression must evaluate to a numeric value for numeric/date attributes and must evaluate to a string for text valued attributes. Can also be an exact character string (not case-sensitive) enclosed in double quotation marks.
ANY	The keyword ANY.
<code>tuple member</code>	Level 0 start tuple (or member) of the independent dimension set. The tuple must contain all the discrete dimensions followed by the continuous dimension members, in the same order that the continuous range has been defined.
<code>tuple member</code>	Optional level 0 end tuple (or member) of the independent dimension set. The tuple must contain all the discrete dimensions followed by the continuous dimension members, in the same order that the continuous range has been defined.

Example

Consider the following scenario: Products are packaged under different ounces over time and the market state, according to the marketing strategy of the company. Ounces is defined as a varying attribute for the Product dimension, to capture the varying attribute association over the continuous Year dimension and the discrete Market dimension.

Year and Market are the independent dimensions, and level-0 tuple months (for example, Jan) combined with a market state (for example, California) is a perspective for which the varying attribute association is defined.

The following query analyzes sales performance of products packaged in units of 20 ounces or greater any time from Jan to Dec in New York, over all quarters. This is the perspective view, which restates the sales according to the packaging strategy in July.

```
WITH PERSPECTIVE (Jul) FOR Ounces
SELECT
  {Qtr1, Qtr2, Qtr3, Qtr4}
ON COLUMNS,
  {WithattrEx(Ounces, ">=", 20, ANY,
    ([New York], Jan), ([New York], Dec))}
ON ROWS
FROM app.db
WHERE
  (Sales, Ounces, [New York])
;
```

See Also

- [AttributeEx](#)

xTD

Returns period-to-date values.

Syntax

```
xTD ( [member ] )
```

Parameter Description

xTD Values:

Parameter	Value
HTD	History-To-Date (H-T-D)
YTD	Year-To-Date
STD	Season-To-Date
PTD	Period-To-Date
QTD	Quarter-To-Date
MTD	Month-To-Date
WTD	Week-To-Date
DTD	Day-To-Date

[member](#) Member specification. Should be a member from the time dimension.

Notes

- xTD ([member]) is equivalent to `PeriodsToDate (layer, [member]`) where *layer* is assumed to be the value set in the corresponding Dynamic Time Series member in the database outline.

For example, in Sample Basic, QTD ([member]) is equivalent to `PeriodsToDate (Year.Generations(2) [,member]`), because Q-T-D is Generation 2 in the Year dimension.

- The xTD functions YTD, QTD, MTD, etc. are not relevant for use in aggregate storage databases, because the xTD functions assume that Dynamic Time Series members are defined in the outline. Dynamic Time Series members are not supported for aggregate storage database outlines.

You can use the [PeriodsToDate](#) function with aggregate storage databases in place of the xTD functions.

For example,

`YTD(May)` is equivalent to `PeriodsToDate(Year.Generations(1), May)`

`QTD(May)` is equivalent to `PeriodsToDate(Year.Generations(2), May)`.

Example

`QTD([Feb])`

returns the set { [Jan], [Feb] }.

`QTD([Feb])` is equivalent to `PeriodsToDate([Year].Generations(2), [Feb])`, because the dynamic-time-series member Q-T-D is defined as Generation 2 of the Year dimension.

`HTD([May])`

returns the set { [Jan], [Feb], [Mar], [Apr], [May] }.

`HTD([May])` is equivalent to `PeriodsToDate([Year].Generations(1), [May])`, because the dynamic-time-series member H-T-D is defined as Generation 1 of the Year dimension.

Note: If a dynamic-time-series member is not defined, an empty set is returned.

`PTD([Feb])`

returns an empty set, because the dynamic-time-series member P-T-D is not enabled in the outline.



Query Logging Configuration

In This Chapter

Query Logging Overview	1161
Query Logging Settings Procedure	1161
Query Log Settings File Syntax	1162
Query Logging Sample File	1165
Query Logging Sample Output	1165

Query Logging Overview

Query logging provides a way for Essbase administrators to track query patterns of an Essbase database. The query log file tracks all queries performed against the database regardless of whether the query originated from Spreadsheet Add-in, an MDX query, or Report Writer. Query logging can track members, generation or level numbers of members belonging to specific generations or levels, and Hybrid Analysis members. Query logging also offers the flexibility to exclude logging of certain dimensions and members belonging to 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.

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 setting to log Hybrid Analysis members for a specified dimension (`QUERYLOG LOGHAMBRS ON | OFF`).
- **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 LOGHAMBRS ON | OFF
 QUERYLOG LOGPATH *path-expression*
 QUERYLOG LOGFORMAT CLUSTER | TUPLE
 QUERYLOG LOGFILESIZE *n*
 QUERYLOG TOTALLOGFILESIZE *n*
 QUERYLOG ON | OFF

QUERYLOG Parameter	Description
[<i>dimension_name</i>]	<p>Identifies the dimension name to be tracked. The brackets around the dimension name are required. QUERYLOG [<i>dimension_name</i>] 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 [<i>dimension_name</i>] setting.</p> <p>Note: QUERYLOG [<i>dimension_name</i>] must precede all settings that track Hybrid Analysis members and members of generation and level ranges; otherwise, Hybrid Analysis and generation and level settings are ignored.</p>
NONE GENERATION <i>generation-range</i>	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 <i>level-range</i>	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 <i>generation-range</i>	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 <i>level-range</i>	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.
LOGHAMBRS ON OFF	Tracks Hybrid Analysis members of the specified dimension. By default, the setting is OFF. The QUERYLOG NONE, GENERATION, and LEVEL parameters do not apply to Hybrid Analysis members because Hybrid Analysis members are not actually members in an Essbase outline. If QUERYLOG LOGHAMBRS ON is set, the log output is always displayed in CLUSTER format, regardless of whether QUERYLOG LOGFORMAT TUPLE is set. If QUERYLOG LOGHAMBRS ON is set, but the database or dimension does not have any Hybrid Analysis members, the setting is ignored.
LOGPATH <i>path-expression</i>	<p>Specifies the location of the output log file. The log file name is <i>dbname00001.qlg</i>; for example, <i>basic00001.qlg</i>. Examples of the log path are QUERYLOG LOGPATH /usr/local/Essbase/logs/ and QUERYLOG LOGPATH d:\Essbase\logs\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.</p> <p>By default, the location for the log output file is the <i>ARBORPATH\app\appname\dbname\</i> directory. If the LOGPATH <i>path-expression</i> setting is missing, the default is used. Essbase writes log information to the query log file after an application stops running.</p>
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.

QUERYLOG Parameter	Description
LOGFILESIZE <i>n</i>	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 <i>n</i>	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:

Generation-Range or Level-Range Value	Description
<i>x</i>	A specific generation or level number. For example, QUERYLOG NONE GENERATION 2 excludes generation 2 from query logging.
<i>x-y</i>	All generations or levels inclusive of number <i>x</i> through number <i>y</i> . For example, QUERYLOG GENERATION 1-3 or QUERYLOG LEVEL 1-3 includes generation or level numbers 1, 2, and 3.
<i>-x</i>	For <i>generation-range</i> , all generations within the range 1 through <i>x</i> . For <i>level-range</i> , all levels within the range 0 through <i>x</i> . For example, QUERYLOG GENERATION -2 includes generations 1 and 2. QUERYLOG LEVEL -3 includes levels 0, 1, 2, and 3.
<i>x-</i>	For <i>generation-range</i> , all generations within the range from number <i>x</i> through the highest generation. For <i>level-range</i> , all levels within the range from number <i>x</i> 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.
- You can place QUERYLOG settings in any order in the settings file, but QUERYLOG [*dimension_name*] must precede any settings that specify Hybrid Analysis members or any setting that specify members by generation or level.
- 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 Hybrid Analysis members of Product, if applicable
QUERYLOG LOGHAMBRS ON
# 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 Outputment 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>
```


9

Report Writer Commands

In This Chapter

Report Writer Overview	1169
Report Writer Syntax	1170
Report Writer Command Groups.....	1171
Examples of Report Scripts	1177
Report Writer Command Reference	1214

Report Writer Overview

Report Writer is a text-based script language that you can use to report on data in multidimensional databases. You can combine Report Writer's selection, layout, and formatting commands to build a variety of reports.

With the Report Writer, you can generate reports whose length or specialized format exceed the capabilities of a spreadsheet. You can use the Report Writer to:

- Define formatted reports on multidimensional data
- Export data from an Essbase database
- Produce free-form reports

To produce reports, Essbase provides several options:

- Use the Report Writer option in Essbase to select commands and options.
- Create a report script using the report editor or any text editor.
- Use a spreadsheet in Essbase template retrieval mode.
- Execute a report script in MaxL or ESSCMD interactive or batch mode.

For an introduction to writing reports, see the *Oracle Essbase Database Administrator's Guide*.

Note: Essbase uses double-precision math as supported by the C compiler on the corresponding platform. Floating point values exceeding the number of significant digits for that platform may result in rounded numbers.

Report Writer Syntax

This topic contains the following information:

- [“Report Delimiters” on page 1170](#)
- [“Syntax Guidelines” on page 1170](#)
- [“Referencing Static Members” on page 1171](#)

Report Delimiters

The < or {} delimiters are required for most Report Writer commands. If you do not use a delimiter, Report Writer assumes that the command name is a member name.

Delimiter	Use in Report Writer:	Example
{}	Encloses report formatting commands	{SUPFORMATS}
<	Precedes layout and member sorting, selection, calculation, and some formatting commands	<PAGE

Syntax Guidelines

- Separate commands with at least one space, tab, or new line. Report processing is not affected by extra blank lines, spaces, or tabs.
- Enter commands in either upper or lowercase. Commands are not case sensitive. If the database outline is case-sensitive, then the member names used in the report script must match the outline.
- To start report processing, enter the ! report output command (exclamation point or "bang"), or one or more consecutive numeric values. You can place one or more report scripts, each terminated by its own ! command, in the same report file.
- You can group more than one format command within a single set of curly braces. For example, these formats are synonyms:

```
{UDATA SKIP}  
{UDATA} {SKIP}
```

- Enclose member names that contain spaces or the member name "Default" in double quotes; for example, "Cost of Goods Sold" "Default".
- If a formatting command is preceded by three or more of the characters "=", "-", and "_," the Report Extractor assumes that the characters are extraneous underline characters and ignores them. For example, ==={SKIP 1}
- Use // (double slash) to indicate a comment. Everything on the line following a comment is ignored by the Report Writer. Each line of a comment must start with a double slash.

Referencing Static Members

You can enter static (non-changing) member names, such as Sales and COGS, directly into the report script. For static member names, use *staticMbrDefinition* syntax, as described below:

Command

A *staticMbrDefinition* specifies the member to select.

Syntax

```
mbrName [ mbrName ]
```

mbrName

Dimension or member name of member to specify. When specifying multiple member names, separate them with spaces. Enclose member names in double quotes if they contain spaces or consist of numbers. For example: "Cost of Goods Sold" or "100-10"

Description

A static member definition specifies a database outline member in a report specification. This definition does not automatically reflect changes to the database outline. If you change a member name in the database outline, you must manually update each report script associated with that outline.

Example

Year

Selects the member Year.

```
Sales "Cost_of_Goods_Sold"
```

Selects the members Sales and Cost_of_Goods_Sold.

Report Writer Command Groups

This section lists all Report Writer commands, grouped by command type. The command groups correspond to the steps of report design:

- [“Report Layout Commands” on page 1172](#)
- [“Data Range Commands” on page 1172](#)
- [“Data Ordering Commands” on page 1172](#)
- [“Member Selection and Sorting Commands” on page 1172](#)
- [“Format Commands” on page 1173](#)
- [“Column or Row Calculation Commands” on page 1176](#)
- [“Member Names and Aliases” on page 1176](#)

For a description of the stages of report design, see the *Oracle Essbase Database Administrator's Guide*.

Report Layout Commands

A report layout is composed of items that make up the columns and rows of a page. Report layout commands provide column, page, and row layout, and include two commands that override the default method for interpreting column dimension member lists. Report Writer provides the following page layout commands:

- [ASYM](#)
- [COLUMN](#)
- [PAGE](#)
- [ROW](#)
- [SYM](#)

Data Range Commands

Data range commands restrict the range of data selected for your reports. Report Writer provides the following data range commands:

- [BOTTOM](#)
- [RESTRICT](#)
- [TOP](#)

Data Ordering Commands

Data ordering commands order data in your reports. Report Writer provides the following ordering command:

- [ORDERBY](#)

Member Selection and Sorting Commands

Member selection commands enhance your selection options using member relationships based on the database outline. The Report Writer provides the following selection and sorting commands:

- [ALLINSAMEDIM](#)
- [ALLSIBLINGS](#)
- [ANCESTORS](#)
- [ATTRIBUTE](#)
- [CHILDREN](#)

- CURRENCY
- DESCENDANTS
- DIMBOTTOM
- DIMEND
- DIMTOP
- DUPLICATE
- IANCESTORS
- ICHILDREN
- IDESCENDANTS
- IPARENT
- LATEST
- LEAVES
- LINK
- MATCH
- OFSAMEGEN
- ONSAMELEVELAS
- PARENT
- SORTALTNAMES
- SORTASC
- SORTDESC
- SORTGEN
- SORTLEVEL
- SORTMBRNames
- SORTNONE
- TODATE
- UDA
- WITHATTR

Format Commands

These commands define the appearance of your data and your report. Each format command applies only to those output lines that *follow* the command.

- ACCON
- ACCOFF
- AFTER
- BEFORE

- BLOCKHEADERS
- BRACKETS
- COLHEADING
- COMMAS
- CURHEADING
- DECIMAL
- ENDHEADING
- EUROPEAN
- FEEDON
- FIXCOLUMNS
- FORMATCOLUMNS
- HEADING
- IMMHEADING
- INCEMPTYROWS
- INCFORMATS
- INCMASK
- INCMISSINGROWS
- INCZEROROWS
- INDENT
- INDENTGEN
- LMARGIN
- MASK
- MISSINGTEXT
- NAMESCOL
- NAMESON
- NAMEWIDTH
- NEWPAGE
- NOINDENTGEN
- NOPAGEONDIMENSION
- NOROWREPEAT
- NOSKIPONDIMENSION
- NOUNAMEONDIM
- ORDER
- OUTALTNAMES
- OUTMBRNames

- OUTPUT
- PAGEHEADING
- PAGELNGTH
- PAGEONDIMENSION
- PYRAMIDHEADERS
- QUOTEMBRNAMES
- RENAME
- ROWREPEAT
- SCALE
- SETCENTER
- SINGLECOLUMN
- SKIP
- SKIPONDIMENSION
- STARTHEADING
- SUPALL
- SUPBRACKETS
- SUPCOLHEADING
- SUPCOMMAS
- SUPCURHEADING
- SUPEMPTYROWS
- SUPEUROPEAN
- SUPFEED
- SUPFORMATS
- SUPHEADING
- SUPMASK
- SUPMISSINGROWS
- SUPNAMES
- SUPOUTPUT
- SUPPAGEHEADING
- SUPSHARE
- SUPSHAREOFF
- SUPZEROROWS
- TABDELIMIT
- TEXT
- UCHARACTERS

- [UCOLUMNS](#)
- [UDATA](#)
- [UNAME](#)
- [UNAMEONDIMENSION](#)
- [UNDERLINECHAR](#)
- [UNDERSCORECHAR](#)
- [WIDTH](#)
- [ZEROTEXT](#)

Column or Row Calculation Commands

These commands perform column and row calculations that let you create extra columns or rows in a report (not defined as part of the database outline) based on selected data members. Enclose all calculation commands and their arguments in curly { } braces.

- [CALCULATE COLUMN](#)
- [CALCULATE ROW](#)
- [CLEARALLROWCALC](#)
- [CLEARROWCALC](#)
- [OFFCOLCALCS](#)
- [OFFROWCALCS](#)
- [ONCOLCALCS](#)
- [ONROWCALCS](#)
- [PRINTROW](#)
- [REMOVECOLCALCS](#)
- [SAVEANDOUTPUT](#)
- [SAVEROW](#)
- [SETROWOP](#)

Member Names and Aliases

These commands allow you to set aliases or alternate names that can make reports easier to read and help your reader focus on the data values rather than the meanings of member (page, column, and row) names.

- [REPALIAS](#)
- [REPALIASMBR](#)
- [REPMBR](#)
- [REPMBRALIAS](#)

- [REPQUALMBR](#)
- [OUTMBRALT](#)
- [OUTALTMBR](#)
- [OUTALT](#)
- [OUTALTNAMES](#)
- [OUTALTSELECT](#)
- [OUTPUTMEMBERKEY](#)

You can use aliases to display members in a report:

- By alias alone. For example, display the name as Diet Cola rather than its corresponding member name 100-20.
- As a combination of member name and alias. For example, display the name as Diet Cola 100-20.

In addition, these report commands also control the display of member names and aliases.

- [ALLINSAMEDIM](#)
- [CHILDREN](#)
- [DESCENDANTS](#)
- [GEN](#)
- [LEV](#)
- [SORTASC](#)
- [SORTALTNAMES](#)
- [SORTDESC](#)
- [SORTGEN](#)
- [SORTLEVEL](#)
- [SORTNONE](#)

Examples of Report Scripts

This section includes report scripts demonstrating report procedures and formats frequently required in business settings.

The samples use both the Demo Basic and Sample Basic databases provided with Essbase Server. The scripts for these examples are available in `\ARBORPATH\App\Demo\Basic` or `\ARBORPATH\App\Sample\Basic`. They are also displayed in Enterprise View in Administration Services, if you chose to install sample applications during installation.

The sample reports demonstrate the following techniques:

- [“Sample 1: Creating a Different Format for Each Page” on page 1178](#)

- [“Sample 2: Handling Missing Values” on page 1179](#)
- [“Sample 3: Nesting Columns” on page 1181](#)
- [“Sample 4: Grouping Rows” on page 1182](#)
- [“Sample 5: Reporting on Different Combinations of Data” on page 1186](#)
- [“Sample 6: Formatting Different Combinations of Data” on page 1187](#)
- [“Sample 7: Using Aliases” on page 1189](#)
- [“Sample 8: Creating Custom Headings and % Characters” on page 1190](#)
- [“Sample 9: Creating Custom Page Headings” on page 1193](#)
- [“Sample 10: Using Formulas” on page 1195](#)
- [“Sample 11: Placing Two-Page Layouts on the Same Page” on page 1196](#)
- [“Sample 12: Formatting for Data Export” on page 1198](#)
- [“Sample 13: Creating Asymmetric Columns” on page 1199](#)
- [“Sample 14: Calculating Columns” on page 1200](#)
- [“Sample 15: Calculating Rows” on page 1202](#)
- [“Sample 16: Sorting by Top or Bottom Data Values” on page 1207](#)
- [“Sample 17: Restricting Rows” on page 1209](#)
- [“Sample 18: Ordering Data Values” on page 1210](#)
- [“Sample 19: Narrowing Member Selection Criteria” on page 1211](#)
- [“Sample 20: Using Attributes in Member Selection” on page 1212](#)
- [“Sample 21: Using the WITHATTR Command in Member Selection” on page 1213](#)

For fundamental information about reports and report scripts, see "Understanding Report Script Basics" in the *Oracle Essbase Database Administrator's Guide*. For detailed information about using Report Writer commands to write reports and reports scripts, see the "Developing Report Scripts" section.

Sample 1: Creating a Different Format for Each Page

This sample report contains data for Actual Sales. Each report page shows a different Product. The report lists products on the same page until the maximum page length is reached. To place each Product on a separate page, you must use the PAGEONDIMENSION format command, as shown in [“Sample 2: Handling Missing Values” on page 1179](#).

Because none of the cities in South sell Stereo or Compact_Disc, the data values indicate #MISSING. You can represent missing values by suppressing the row or substituting a replacement text string, such as N/A. See [“Sample 2: Handling Missing Values” on page 1179](#) for an example of substituting page breaks and labels for missing values.

Sales Actual Stereo				
	Qtr1	Qtr2	Qtr3	Qtr4

	=====	=====	=====	=====
East	7,839	7,933	7,673	10,044
West	11,633	11,191	11,299	14,018
South	#Missing	#Missing	#Missing	#Missing
Market	19,472	19,124	18,972	24,062

Sales Actual Compact_Disc

	Qtr1	Qtr2	Qtr3	Qtr4
=====	=====	=====	=====	=====
East	10,293	9,702	9,965	11,792
West	14,321	14,016	14,328	17,247
South	#Missing	#Missing	#Missing	#Missing
Market	24,614	23,718	24,293	29,039

Sales Actual Audio

	Qtr1	Qtr2	Qtr3	Qtr4
=====	=====	=====	=====	=====
East	18,132	17,635	17,638	21,836
West	25,954	25,207	25,627	31,265
South	#Missing	#Missing	#Missing	#Missing
Market	44,086	42,842	43,265	53,101

Use the following script to create Sample 1:

```
<PAGE (Accounts, Scenario, Product)
Sales
Actual
<IDESCENDANTS Audio

    <COLUMN (Year)
    <CHILDREN Year

<ROW(Market)
<ICHILDREN Market
!
```

The ! report output command is required to generate the report.

Because the IDESCENDANTS selection command is used for Audio, the report selects all three members. Only a single member is selected from the other page dimensions, Sales and Actual. As a result, the script creates three report pages. They display as one long report page unless you use the PAGEONDIMENSION format command, as shown in [“Sample 2: Handling Missing Values” on page 1179](#).

This report script, ACTSALES.REP, is available in the \ARBORPATH\App\Demo\Basic directory, and is displayed in Enterprise View in Administration Services.

Sample 2: Handling Missing Values

This report has the same layout and member selection as Sample 1, and shows you how to use page breaks and labels for missing values.

Sales Actual Stereo

	Qtr1	Qtr2	Qtr3	Qtr4
	=====	=====	=====	=====
East	7,839	7,933	7,673	10,044
West	11,633	11,191	11,299	14,018
South	N/A	N/A	N/A	N/A
Market	19,472	19,124	18,972	24,062

Sales Actual Compact_Disc

	Qtr1	Qtr2	Qtr3	Qtr4
	=====	=====	=====	=====
East	10,293	9,702	9,965	11,792
West	14,321	14,016	14,328	17,247
South	N/A	N/A	N/A	N/A
Market	24,614	23,718	24,293	29,039

Sales Actual Audio

	Qtr1	Qtr2	Qtr3	Qtr4
	=====	=====	=====	=====
East	18,132	17,635	17,638	21,836
West	25,954	25,207	25,627	31,265
South	N/A	N/A	N/A	N/A
Market	44,086	42,842	43,265	53,101

Use the following script to create Sample 2:

```
<PAGE (Accounts, Scenario, Product)
Sales
Actual
<IDESCENDANTS Product
{ PAGEONDIMENSION Product }
{ MISSINGTEXT "N/A" }

    <COLUMN (Year)
    <CHILDREN Year

<ROW(Market)
<ICHILDREN Market
!
```

The PAGEONDIMENSION format command creates a page break whenever a member from the specified dimension changes. Because the report selects eight Product members, the report is eight pages long.

The MISSINGTEXT format command substitutes any strings enclosed within double quotes into the #MISSING string. To suppress missing values, use the SUPMISSINGROWS command.

You can also combine format commands within one set of braces:

```
{ PAGEONDIMENSION Product MISSINGTEXT "N/A" }
```

This report script, MISS_LBL.REP, is available in the \ARBORPATH\App\Demo\Basic directory, and is displayed in Enterprise View in Administration Services.

Sample 3: Nesting Columns

Each page produced by this report sample contains Sales information for a given Market. The report has two groups of columns across the page. The Actual and Budget members are the nested column group below Year members.

Note that the Actual and Budget members are on the same line in the report. You can put multiple commands on one line, but report commands are easier to read if they are spread out.

Sales East

	Jan		Feb		Mar		Qtr1	
	Actual	Budget	Actual	Budget	Actual	Budget	Actual	Budget
	=====	=====	=====	=====	=====	=====	=====	=====
Stereo	2,788	2,950	2,482	2,700	2,569	2,700	7,839	8,350
Compact_Disc	3,550	3,450	3,285	3,250	3,458	3,250	10,293	9,950
Audio	6,338	6,400	5,767	5,950	6,027	5,950	18,132	18,300
Television	5,244	4,800	4,200	4,300	3,960	4,300	13,404	13,400
VCR	4,311	4,200	3,734	3,700	3,676	3,700	11,721	11,600
Camera	2,656	2,850	2,525	2,670	2,541	2,670	7,722	8,190
Visual	12,211	11,850	10,459	10,670	10,177	10,670	32,847	33,190
Product	18,549	18,250	16,226	16,620	16,204	16,620	50,979	51,490

Sales West

	Jan		Feb		Mar		Qtr1	
	Actual	Budget	Actual	Budget	Actual	Budget	Actual	Budget
	=====	=====	=====	=====	=====	=====	=====	=====
Stereo	4,102	4,000	3,723	3,600	3,808	3,600	11,633	11,200
Compact_Disc	4,886	4,700	4,647	4,400	4,788	4,400	14,321	13,500
Audio	8,988	8,700	8,370	8,000	8,596	8,000	25,954	24,700
Television	5,206	5,100	4,640	4,600	4,783	4,600	14,629	14,300
VCR	4,670	4,650	4,667	4,200	4,517	4,200	13,854	13,050
Camera	3,815	4,050	3,463	3,750	3,478	3,750	10,756	11,550
Visual	13,691	13,800	12,770	12,550	12,778	12,550	39,239	38,900
Product	22,679	22,500	21,140	20,550	21,374	20,550	65,193	63,600

/pre>

Sales South

	Jan		Feb		Mar		Qtr1	
	Actual	Budget	Actual	Budget	Actual	Budget	Actual	Budget
	=====	=====	=====	=====	=====	=====	=====	=====
Television	3,137	3,400	2,929	3,100	2,815	3,100	8,881	9,600
VCR	3,225	3,400	3,206	3,100	3,120	3,100	9,551	9,600
Camera	2,306	2,400	2,167	2,400	2,168	2,400	6,641	7,200
Visual	8,668	9,200	8,302	8,600	8,103	8,600	25,073	26,400
Product	8,668	9,200	8,302	8,600	8,103	8,600	25,073	26,400

Sales Market

	Jan		Feb		Mar		Qtr1	
	Actual	Budget	Actual	Budget	Actual	Budget	Actual	Budget
	=====	=====	=====	=====	=====	=====	=====	=====

Stereo	6,890	6,950	6,205	6,300	6,377	6,300	19,472	19,550
Compact_Disc	8,436	8,150	7,932	7,650	8,246	7,650	24,614	23,450
Audio	15,326	15,100	14,137	13,950	14,623	13,950	44,086	43,000
Television	13,587	13,300	11,769	12,000	11,558	12,000	36,914	37,300
VCR	12,206	12,250	11,607	11,000	11,313	11,000	35,126	34,250
Camera	8,777	9,300	8,155	8,820	8,187	8,820	25,119	26,940
Visual	34,570	34,850	31,531	31,820	31,058	31,820	97,159	98,490
Product	49,896	49,950	45,668	45,770	45,681	45,770	141,245	141,490

Use the following script to create Sample 3:

```
<PAGE (Accounts, Market)
Sales
<ICHILDREN Market
{ PAGEONDIMENSION Market }
{ SUPMISSINGROWS }
  <COLUMN (Year, Scenario)
  <ICHILDREN Qtr1
  Actual Budget
<ROW(Product)
<IDESCENDANTS Product
!
```

The report selects four Markets because the <ICHILDREN command is applied to Market. Only Sales is selected from the other page dimension, so the report has four pages.

For the South, all the rows of Product data are not displayed. Recall that the cities in the South do not sell every Product. The report uses the SUPMISSINGROWS format command to suppress the output of any member rows with all missing values.

This report script, COLGROUP.REP, is available in the \ARBORPATH\App\Demo\Basic directory, and is displayed in Enterprise View in Administration Services.

Sample 4: Grouping Rows

Each page of this report contains Sales information for a given Market. The report page contains members for both Product and Year as groups of rows down the page. This script creates a four-page report because the page dimensions and their member selections are the same as in “[Sample 3: Nesting Columns](#)” on page 1181. The row and column layout is switched because the row and column dimensions are different. This section shows a representative part of the output.

		Sales East		
		Actual	Budget	Variance
		=====	=====	=====
Stereo	Qtr1	7,839	8,350	(511)
	Qtr2	7,933	8,150	(217)
	Qtr3	7,673	8,350	(677)
	Qtr4	10,044	10,400	(356)
	Year	33,489	35,250	(1,761)
Compact_Disc	Qtr1	10,293	9,950	343
	Qtr2	9,702	9,750	(48)
	Qtr3	9,965	10,050	(85)
	Qtr4	11,792	12,550	(758)
	Year	41,752	42,300	(548)

Audio	Qtr1	18,132	18,300	(168)
	Qtr2	17,635	17,900	(265)
	Qtr3	17,638	18,400	(762)
	Qtr4	21,836	22,950	(1,114)
	Year	75,241	77,550	(2,309)
Television	Qtr1	13,404	13,400	4
	Qtr2	12,115	12,900	(785)
	Qtr3	15,014	14,200	814
	Qtr4	17,861	17,300	561
	Year	58,394	57,800	594
VCR	Qtr1	11,721	11,600	121
	Qtr2	10,999	11,100	(101)
	Qtr3	13,217	11,800	1,417
	Qtr4	14,386	14,900	(514)
	Year	50,323	49,400	923
Camera	Qtr1	7,722	8,190	(468)
	Qtr2	7,581	8,210	(629)
	Qtr3	8,181	8,630	(449)
	Qtr4	10,853	11,550	(697)
	Year	34,337	36,580	(2,243)
Visual	Qtr1	32,847	33,190	(343)
	Qtr2	30,695	32,210	(1,515)
	Qtr3	36,412	34,630	1,782
	Qtr4	43,100	43,750	(650)
	Year	143,054	143,780	(726)
Product	Qtr1	50,979	51,490	(511)
	Qtr2	48,330	50,110	(1,780)
	Qtr3	54,050	53,030	1,020
	Qtr4	64,936	66,700	(1,764)
	Year	218,295	221,330	(3,035)

		Sales West		
		Actual	Budget	Variance
		=====	=====	=====
Stereo	Qtr1	11,633	11,200	433
	Qtr2	11,191	11,050	141
	Qtr3	11,299	11,650	(351)
	Qtr4	14,018	14,500	(482)
	Year	48,141	48,400	(259)
Compact_Disc	Qtr1	14,321	13,500	821
	Qtr2	14,016	13,500	516
	Qtr3	14,328	14,300	28
	Qtr4	17,247	16,700	547
	Year	59,912	58,000	1,912
Audio	Qtr1	25,954	24,700	1,254
	Qtr2	25,207	24,550	657
	Qtr3	25,627	25,950	(323)
	Qtr4	31,265	31,200	65
	Year	108,053	106,400	1,653
Television	Qtr1	14,629	14,300	329
	Qtr2	14,486	13,800	686
	Qtr3	14,580	14,000	580
	Qtr4	20,814	19,400	1,414
	Year	64,509	61,500	3,009
VCR	Qtr1	13,854	13,050	804
	Qtr2	13,156	12,600	556
	Qtr3	15,030	13,750	1,280

	Qtr4	18,723	17,950	773
	Year	60,763	57,350	3,413
Camera	Qtr1	10,756	11,550	(794)
	Qtr2	10,573	11,400	(827)
	Qtr3	10,735	11,550	(815)
	Qtr4	13,906	15,000	(1,094)
	Year	45,970	49,500	(3,530)
Visual	Qtr1	39,239	38,900	339
	Qtr2	38,215	37,800	415
	Qtr3	40,345	39,300	1,045
	Qtr4	53,443	52,350	1,093
	Year	171,242	168,350	2,892
Product	Qtr1	65,193	63,600	1,593
	Qtr2	63,422	62,350	1,072
	Qtr3	65,972	65,250	722
	Qtr4	84,708	83,550	1,158
	Year	279,295	274,750	4,545

/pre>

Sales South

		Actual	Budget	Variance
		=====	=====	=====
Television	Qtr1	8,881	9,600	(719)
	Qtr2	8,627	9,300	(673)
	Qtr3	8,674	9,300	(626)
	Qtr4	12,919	12,600	319
	Year	39,101	40,800	(1,699)
VCR	Qtr1	9,551	9,600	(49)
	Qtr2	9,049	9,300	(251)
	Qtr3	9,998	10,000	(2)
	Qtr4	12,923	13,600	(677)
	Year	41,521	42,500	(979)
Camera	Qtr1	6,641	7,200	(559)
	Qtr2	6,765	7,350	(585)
	Qtr3	6,798	7,500	(702)
	Qtr4	9,486	10,200	(714)
	Year	29,690	32,250	(2,560)
Visual	Qtr1	25,073	26,400	(1,327)
	Qtr2	24,441	25,950	(1,509)
	Qtr3	25,470	26,800	(1,330)
	Qtr4	35,328	36,400	(1,072)
	Year	110,312	115,550	(5,238)
Product	Qtr1	25,073	26,400	(1,327)
	Qtr2	24,441	25,950	(1,509)
	Qtr3	25,470	26,800	(1,330)
	Qtr4	35,328	36,400	(1,072)
	Year	110,312	115,550	(5,238)

Sales Market

		Actual	Budget	Variance
		=====	=====	=====
Stereo	Qtr1	19,472	19,550	(78)
	Qtr2	19,124	19,200	(76)
	Qtr3	18,972	20,000	(1,028)

	Qtr4	24,062	24,900	(838)
	Year	81,630	83,650	(2,020)
Compact_Disc	Qtr1	24,614	23,450	1,164
	Qtr2	23,718	23,250	468
	Qtr3	24,293	24,350	(57)
	Qtr4	29,039	29,250	(211)
	Year	101,664	100,300	1,364
Audio	Qtr1	44,086	43,000	1,086
	Qtr2	42,842	42,450	392
	Qtr3	43,265	44,350	(1,085)
	Qtr4	53,101	54,150	(1,049)
	Year	183,294	183,950	(656)
Television	Qtr1	36,914	37,300	(386)
	Qtr2	35,228	36,000	(772)
	Qtr3	38,268	37,500	768
	Qtr4	51,594	49,300	2,294
	Year	162,004	160,100	1,904
VCR	Qtr1	35,126	34,250	876
	Qtr2	33,204	33,000	204
	Qtr3	38,245	35,550	2,695
	Qtr4	46,032	46,450	(418)
	Year	152,607	149,250	3,357
Camera	Qtr1	25,119	26,940	(1,821)
	Qtr2	24,919	26,960	(2,041)
	Qtr3	25,714	27,680	(1,966)
	Qtr4	34,245	36,750	(2,505)
	Year	109,997	118,330	(8,333)
Visual	Qtr1	97,159	98,490	(1,331)
	Qtr2	93,351	95,960	(2,609)
	Qtr3	102,227	100,730	1,497
	Qtr4	131,871	132,500	(629)
	Year	424,608	427,680	(3,072)
Product	Qtr1	141,245	141,490	(245)
	Qtr2	136,193	138,410	(2,217)
	Qtr3	145,492	145,080	412
	Qtr4	184,972	186,650	(1,678)
	Year	607,902	611,630	(3,728)

Use the following script to create Sample 4:

```

<PAGE (Accounts, Market)
Sales
<ICHILDREN Market
{ PAGEONDIMENSION Market }
{ SUPMISSINGROWS }

        <COLUMN (Scenario)
        <CHILDREN Scenario

<ROW(Product3, Year)
<ICHILDREN Year
<IDESCENDANTS Product
!
```

This report script, ROWGROUP.REP, is available in the \ARBORPATH\App\Demo\Basic directory, and is displayed in Enterprise View in Administration Services.

Sample 5: Reporting on Different Combinations of Data

Each page represents a different combination of Product, Market, and Budget data. The total number of pages is determined by the number of Market and Product members. This section shows a representative part of the output.

Some data values have four decimal places. The number of decimal places, by default, is output to the true number of decimal values of the data cell. “[Sample 6: Formatting Different Combinations of Data](#)” on page 1187 uses the DECIMAL format command to define a specific number of places.

The member selection commands select three Product members and fourteen Market members, producing a 42-page report. The number of report pages is determined by multiplying the number of members selected from each page dimension.

Budget Audio New_York					
	Qtr1	Qtr2	Qtr3	Qtr4	Year
	=====	=====	=====	=====	=====
Sales	6,400	6,400	6,700	8,350	27,850
Cost_of_Goods_Sold	3,012	3,012	3,146	3,973	13,143
Margin	3,388	3,388	3,554	4,377	14,707
Marketing	525	515	475	555	2,070
Payroll	1,950	1,950	1,950	1,950	7,800
Miscellaneous	0	0	0	0	0
Total_Expenses	2,475	2,465	2,425	2,505	9,870
Profit	913	923	1,129	1,872	4,837
Profit_%	14	14	17	22	17
Margin_%	53	53	53	52	53

Budget Audio Boston					
	Qtr1	Qtr2	Qtr3	Qtr4	Year
	=====	=====	=====	=====	=====
Sales	6,050	5,750	5,900	7,350	25,050
Cost_of_Goods_Sold	2,829	2,695	2,762	3,413	11,699
Margin	3,221	3,055	3,138	3,937	13,351
Marketing	410	400	400	520	1,730
Payroll	1,590	1,590	1,590	1,590	6,360
Miscellaneous	0	0	0	0	0
Total_Expenses	2,000	1,990	1,990	2,110	8,090
Profit	1,221	1,065	1,148	1,827	5,261
Profit_%	20	19	19	25	21
Margin_%	53	53	53	54	53

Budget Product Market					
	Qtr1	Qtr2	Qtr3	Qtr4	Year
	=====	=====	=====	=====	=====
Sales	141,490	138,410	145,080	186,650	611,630
Cost_of_Goods_Sold	55,860	54,579	57,379	73,276	241,093
Margin	85,630	83,831	87,702	113,374	370,537
Marketing	10,555	10,680	10,780	13,915	45,930
Payroll	43,234	43,248	43,248	43,248	172,978

Miscellaneous	0	0	0	0	0
Total_Expenses	53,789	53,928	54,028	57,163	218,908
Profit	31,841	29,903	33,674	56,211	151,629
Profit_%	23	22	23	30	25
Margin_%	61	61	60	61	61

Use the following script to create Sample 5:

```

<PAGE (Scenario, Product, Market)
Budget
<CHILDREN Product
<DESCENDANTS Market
{ PAGEONDIMENSION Product } // New page at each new Product
{ PAGEONDIMENSION Market } // New page at each new Market
  <COLUMN (Year)
  <CHILDREN Year

<ROW(Accounts)
<DESCENDANTS Accounts
  !

```

This report script, COMBO1.REP, is available in the \ARBORPATH\App\Demo\Basic directory, and is displayed in Enterprise View in Administration Services.

Sample 6: Formatting Different Combinations of Data

This report uses the same layout and member selection as Sample 5, and adds more formatting in the report body. Note the use of line formatting.

	Budget Audio New_York				
	Qtr1	Qtr2	Qtr3	Qtr4	Year
	=====	=====	=====	=====	=====
Sales	6,400	6,400	6,700	8,350	27,850
Cost_of_Goods_Sold	3,012	3,012	3,146	3,973	13,143
Margin	3,388	3,388	3,554	4,377	14,707
Marketing	525	515	475	555	2,070
Payroll	1,950	1,950	1,950	1,950	7,800
Miscellaneous	0	0	0	0	0
Total_Expenses	2,475	2,465	2,425	2,505	9,870
Profit	913	923	1,129	1,872	4,837
Profit_%	14.27	14.42	16.85	22.42	17.37
Margin_%	52.94	52.94	53.04	52.42	52.81

	Budget Audio Boston				
	Qtr1	Qtr2	Qtr3	Qtr4	Year
	=====	=====	=====	=====	=====

Sales	6,050	5,750	5,900	7,350	25,050
Cost_of_Goods_Sold	2,829	2,695	2,762	3,413	11,699
Margin	3,221	3,055	3,138	3,937	13,351
Marketing	410	400	400	520	1,730
Payroll	1,590	1,590	1,590	1,590	6,360
Miscellaneous	0	0	0	0	0
Total_Expenses	2,000	1,990	1,990	2,110	8,090
Profit	1,221	1,065	1,148	1,827	5,261
Profit_%	20.18	18.52	19.46	24.86	21.00
Margin_%	53.24	53.13	53.19	53.56	53.30

Use the following script to create Sample 6:

```

<PAGE (Scenario, Product, Market)
{ PAGEONDIMENSION Product PAGEONDIMENSION Market }
Budget
<ICHILDREN Product
<IDESCENDANTS Market
    <COLUMN (Year)
    <ICHILDREN Year
<ROW(Accounts)
{ SUPBRACKETS DECIMAL 0 }
Sales
Cost_of_Goods_Sold
{ UDATA "-" } //line formatting command
Margin
{ SKIP }
Marketing
Payroll
Miscellaneous
{ UDATA "-" } //line formatting command
Total_Expenses
{ SKIP }
Profit
{ UDATA DECIMAL 2 } //line formatting command
Profit_%
Margin_%
!
```

Format commands apply to members that follow the commands. The report begins each new page with the formats in place at the end of the previous report page. For example, if a report page ends with two decimal places, the following page begins with two decimal places. This report demonstrates the use of several important format commands:

- DECIMAL-The script for this report specifies the DECIMAL 0 format command before the Sales member.
- SUPBRACKETS-By default, negative numbers are enclosed in brackets, (). The SUPBRACKETS format command causes negative numbers to be output with a minus sign.
- UDATA-The UDATA command places underline characters under data columns. The character is specified within double quotes. The default is a double underline.

This report script, COMBO2.REP, is available in the \ARBORPATH\App\Demo\Basic directory, and is displayed in Enterprise View in Administration Services.

Sample 7: Using Aliases

This report outputs members in the middle of a page and uses aliases or alternate names. The default row member indentation is turned off.

Stereo Market				
Qtr4			Year	
Actual	Budget		Actual	Budget
=====	=====		=====	=====
24,062	24,900	Sales	81,630	83,650
13,937	14,442	COGS	47,654	48,517

10,125	10,458	Margin	33,976	35,133
1,438	1,600	Marketing	4,933	5,465
7,110	6,840	Payroll	28,440	27,360
-200	0	Misc.	-143	0

8,348	8,440	Total_Expenses	33,230	32,825
1,777	2,018	Profit	746	2,308
=====	=====		=====	=====
7.39	8.10	Profit_%	0.91	2.76
42.08	42.00	Margin_%	41.62	42.00

Compact_Disc Market				
Qtr4			Period	
Actual	Budget		Actual	Budget
=====	=====		=====	=====
29,039	29,250	Sales	101,664	100,300
10,830	11,115	COGS	38,120	38,114

18,209	18,135	Margin	63,544	62,186
1,669	1,780	Marketing	6,067	5,975
5,721	5,415	Payroll	22,200	21,660
-226	0	Misc.	97	0

7,164	7,195	Total_Expenses	28,364	27,635
11,045	10,940	Profit	35,180	34,551
=====	=====		=====	=====
38.04	37.40	Profit_%	34.60	34.45
62.71	62.00	Margin_%	62.50	62.00

Use the following script to create Sample 7:

```
<PAGE (Product, Market)
{ PAGEONDIMENSION Product }
{ PAGEONDIMENSION Market }
```

```

<IDESCENDANTS Product
{ DECIMAL 0 }
<SYM

    <COLUMN (Year, Scenario)
    Qtr4 Year
    Actual Budget
<ROW(Accounts)
{ SUPBRACKETS OUTALTNAMES NOINDENTGEN ORDER 1,2,0,3,4 }
Sales Cost_of_Goods_Sold
{ UDATA "-" }
Margin
{ SKIP }
Marketing Payroll Miscellaneous
{ UDATA "-" }
Total_Expenses
{ SKIP }
Profit
{ UDATA DECIMAL 2 }
Profit_%
Margin_%
!
```

The SYM command forces the report to output symmetric column groups. The default is to display two columns—one for Qtr4 Actual and one for Year Budget. Because the report calls for Actual and Budget under both Qtr4 and Year, the SYM command is required. Alternatively, repeat the Actual and Budget names under Qtr4 and Year.

The OUTALTNAMES format command causes the report to use aliases or alternate names instead of member names.

The NOINDENTGEN format command causes row members to not be indented. By default, members are indented two spaces for each level.

The ORDER command moves specified output columns to new locations. The row name is considered column 0.

The FIXCOLUMNS format command restricts the number of output columns. Reports often require both ORDER and FIXCOLUMNS. You can use ORDER to remove unwanted columns, and FIXCOLUMNS to stop these columns from displaying after the report columns.

This report script, MIDDLE.REP, is available in the \ARBORPATH\App\Demo\Basic directory, and is displayed in Enterprise View in Administration Services.

Sample 8: Creating Custom Headings and % Characters

This report displays custom headings and percent sign (%) characters after each data value. This section shows a representative part of the output.

```

Prepared by: Admin                The Electronics Club                Page: 1
                                                                         09/21/01

                                     Profit_% Actual Stereo

                                     Jan      Feb      Mar      Apr      May      Jun
```

	=====	=====	=====	=====	=====	=====
New_York	1.43%	-10.00%	-3.51%	-2.22%	1.14%	-6.18%
Boston	-0.34%	-2.51%	-4.44%	-4.89%	-7.02%	-13.15%
Chicago	-0.65%	-0.72%	-2.28%	-3.53%	-6.33%	-10.79%
East	0.18%	-4.47%	-3.39%	-3.41%	-3.60%	-9.70%
San_Francisco	1.43%	-1.87%	4.42%	2.15%	-1.26%	0.66%
Seattle	0.95%	-5.66%	1.42%	-6.82%	-11.47%	-12.34%
Denver	3.03%	-1.11%	-5.88%	-6.52%	-5.17%	-13.83%
Los_Angeles	-1.50%	-3.94%	-2.86%	-3.29%	3.12%	-2.51%
West	0.98%	-2.95%	-0.13%	-2.81%	-2.62%	-5.61%
Dallas	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Houston	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Phoenix	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
South	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Market	0.65%	-3.56%	-1.44%	-3.06%	-3.03%	-7.29%

Prepared by: Admin

The Electronics Club

Page: 2
09/21/01

Profit_% Actual Compact_Disc

	Jan	Feb	Mar	Apr	May	Jun
	=====	=====	=====	=====	=====	=====
New_York	32.51%	29.95%	35.30%	32.70%	30.45%	31.73%
Boston	33.42%	27.92%	33.98%	30.74%	27.45%	30.85%
Chicago	34.29%	30.48%	26.33%	28.83%	28.11%	33.76%
East	33.35%	29.50%	32.30%	30.92%	28.77%	32.09%
San_Francisco	37.77%	35.02%	33.41%	33.23%	35.32%	37.95%
Seattle	40.41%	38.33%	38.89%	37.06%	37.01%	38.29%
Denver	31.93%	32.10%	34.82%	29.15%	32.71%	30.85%
Los_Angeles	31.65%	30.22%	30.22%	31.45%	27.06%	33.20%
West	35.51%	33.94%	34.21%	32.77%	33.16%	35.25%
Dallas	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Houston	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Phoenix	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
South	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Market	34.60%	32.10%	33.41%	32.01%	31.35%	33.97%

Prepared by: Admin

The Electronics Club

Page: 3
09/21/01

Profit_% Actual Audio

	Jan	Feb	Mar	Apr	May	Jun
	=====	=====	=====	=====	=====	=====
New_York	19.35%	13.64%	18.64%	16.55%	16.70%	14.65%
Boston	18.34%	14.44%	18.94%	14.94%	12.14%	12.42%
Chicago	18.50%	16.67%	13.18%	14.12%	12.70%	13.74%
East	18.76%	14.88%	17.09%	15.32%	14.05%	13.68%
San_Francisco	20.32%	17.38%	18.92%	18.03%	18.23%	20.57%
Seattle	23.36%	21.40%	23.37%	20.17%	18.82%	19.04%
Denver	18.36%	17.25%	18.88%	13.43%	15.84%	12.14%
Los_Angeles	17.15%	14.76%	15.44%	15.76%	15.10%	17.07%
West	19.75%	17.53%	19.00%	16.88%	17.01%	17.52%
Dallas	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%

Houston	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Phoenix	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
South	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Market	19.34%	16.45%	18.21%	16.24%	15.78%	15.96%

Prepared by: Admin The Electronics Club Page: 8
09/21/01

Profit_% Actual Product

	Jan	Feb	Mar	Apr	May	Jun
	=====	=====	=====	=====	=====	=====
New_York	22.71%	21.43%	13.11%	10.54%	9.73%	13.16%
Boston	24.98%	23.25%	19.95%	18.00%	17.03%	18.62%
Chicago	22.01%	17.94%	18.14%	15.45%	18.70%	16.01%
East	23.19%	20.84%	16.89%	14.42%	14.94%	15.78%
San_Francisco	23.71%	20.60%	21.93%	20.45%	21.44%	19.98%
Seattle	21.06%	21.05%	21.24%	19.00%	21.72%	15.13%
Denver	21.61%	16.01%	19.79%	14.81%	20.66%	13.89%
Los_Angeles	17.54%	15.51%	17.03%	14.33%	17.59%	16.09%
West	21.02%	18.35%	19.99%	17.26%	20.30%	16.61%
Dallas	15.67%	16.50%	15.32%	13.93%	20.36%	15.49%
Houston	20.01%	20.29%	20.62%	15.87%	23.60%	12.38%
Phoenix	20.01%	16.12%	17.18%	16.50%	21.39%	15.22%
South	18.39%	17.53%	17.59%	15.36%	21.66%	14.46%
Market	21.37%	19.09%	18.46%	15.92%	18.67%	15.93%

Use the following script to create Sample 8:

```
<PAGE (Accounts, Scenario, Product)
{ PAGEONDIMENSION Product } // New page when Product changes
Profit_%
Actual
<IDESCENDANTS Product

      <COLUMN (Year)
      Jan Feb Mar Apr May Jun

<ROW(Market)

{ STARTHEADING
TEXT  1 "Prepared by:"
      14 "**USERNAME"
      C "The Electronics Club"
      65 "**PAGESTRING"
TEXT  65 "**DATE"
SKIP
ENDHEADING }

{ Decimal 2 AFTER "%" SUPBRACKETS } // Place % at end and
// suppress bracket
<IDESCENDANTS Market
!
```

Each data value in the report has a percent sign, %. This label is defined with the AFTER "%" format command. You can specify any character within quotation marks.

Los Angeles	31.65%	30.22%	30.22%	31.45%	27.06%	33.20%
Dallas	#Missing	#Missing	#Missing	#Missing	#Missing	#Missing
Houston	#Missing	#Missing	#Missing	#Missing	#Missing	#Missing
Phoenix	#Missing	#Missing	#Missing	#Missing	#Missing	#Missing
East	33.35%	29.50%	32.30%	30.92%	28.77%	32.09%
West	35.51%	33.94%	34.21%	32.77%	33.16%	35.25%
South	#Missing	#Missing	#Missing	#Missing	#Missing	#Missing
Market	34.60%	32.10%	33.41%	32.01%	31.35%	33.97%

Prepared by :admin

The Electronics Club
Actual Profit by Product

Page: 8
12/12/01

Product:Product

	Jan	Feb	Mar	Apr	May	Jun
New York	22.71%	21.43%	13.11%	10.54%	9.73%	13.16%
Boston	24.98%	23.25%	19.95%	18.00%	17.03%	18.62%
Chicago	22.01%	17.94%	18.14%	15.45%	18.70%	16.01%
San Francisco	23.71%	20.60%	21.93%	20.45%	21.44%	19.98%
Seattle	21.06%	21.05%	21.24%	19.00%	21.72%	15.13%
Denver	21.61%	16.01%	19.79%	14.81%	20.66%	13.89%
Los Angeles	17.54%	15.51%	17.03%	14.33%	17.59%	16.09%
Dallas	15.67%	16.50%	15.32%	13.93%	20.36%	15.49%
Houston	20.01%	20.29%	20.62%	15.87%	23.60%	12.38%
Phoenix	20.01%	16.12%	17.18%	16.50%	21.39%	15.22%
East	23.19%	20.84%	16.89%	14.42%	14.94%	15.78%
West	21.02%	18.35%	19.99%	17.26%	20.30%	16.61%
South	18.39%	17.53%	17.59%	15.36%	21.66%	14.46%
Market	21.37%	19.09%	18.46%	15.92%	18.67%	15.93%

Use the following script to create Sample 9:

```

<PAGE (Accounts, Scenario, Product)
<IDESCENDANTS Product
<SORTLEVEL
{ PAGEONDIMENSION Product }
{ STARTHEADING
TEXT      1 "Prepared by:"
          14 "**USERNAME"
           C "The Electronics Club"
          65 "**PAGESTRING"
SUPPAGEHEADING
UNDERLINECHAR " "
TEXT      C "Actual Profit by Product"
          65 "**DATE"
TEXT      1 "Product:"
          10 "**PAGEHDR 3"
SKIP
ENDHEADING }
Profit_%
Actual

          <COLUMN (Year)
          Jan Feb Mar Apr May Jun
<ROW(Market)

{ DECIMAL 2 AFTER "%" SUPBRACKETS UNDERSCORECHAR " " }
{ INDENTGEN 1 }

```

```
<IDESCENDANTS Market
!
```

The SUPPAGEHEADING format command suppresses the default page headings from output.

The *PAGEHDR command customizes the location of page member labels. The Sample 9 script uses page heading number 3, Product because this is the third page dimension.

You may have also noticed that member names do not have underscores. The UNDERSCORECHAR format command blanks out underscores.

Another difference is the underlining of column headings. The UNDERLINECHAR format command causes the underlining character to change to the character in quotes.

The report rows are also sorted according to their levels in the database outline. Sort commands, such as SORTLEVEL, do not affect individual members selected in reports. Instead, these commands work in conjunction with member selection commands.

Note: You can use only one sort command in a report.

Sample 9 reverses the indentation of levels from previous reports. The INDENTGEN command indents members to the specified number of characters.

This report script, HEADING2.REP, is available in the \ARBORPATH\App\Demo\Basic directory, and is displayed in Enterprise View in Administration Services.

Sample 10: Using Formulas

Column calculation formulas manipulate the column value of a particular row or a constant. In this report sample, each % column represents the quarterly values as a percent of Sales for the respective quarter. In addition, the Avg column represents an average value for the two quarters.

	Actual Product Market				
	Qtr1	%	Qtr2	%	Avg
	=====	=====	=====	=====	=====
Sales	141,245	100.00	136,193	100.00	138,719
Cost_of_Goods_Sold	58,104	41.14	56,281	41.32	57,193
Margin	83,141	58.86	79,912	58.68	81,527
Marketing	11,211	7.94	11,302	8.30	11,257
Payroll	43,817	31.02	43,827	32.18	43,822
Miscellaneous	302	0.21	1,859	1.36	1,081
Total_Expenses	55,330	39.17	56,988	41.84	56,159
Profit	27,811	19.69	22,924	16.83	25,368
Profit_%	20	0.01	17	0.01	18
Margin_%	59	0.04	59	0.04	59

Use the following script to create Sample 10:

```
// This report performs column calculations based on values in a
// report row.
```

```
<PAGE (Scenario, Product, Market)
```

Actual

```
<COLUMN (Year)
  Qtr1 Qtr2

{ DECIMAL 2 3 4 }
{ NAMEWIDTH 22 WIDTH 7 3 4 }
{ ORDER 0 1 3 2 4 5 }

<ROW (Accounts)
{ SAVEROW } Sales
  !

{ CALCULATE COLUMN "%" = 1 % "Sales" 1 }
{ CALCULATE COLUMN "% " = 2 % "Sales" 2 }
{ CALCULATE COLUMN "Avg" = 1 + 2 / 2. }

<DESCENDANTS Accounts
  !
```

Note: You can include comments in the report by preceding the text with //. The Report Extractor ignores everything that follows the double slash. You can use comments to explain report processing.

The SAVEROW command reserves space for a row member that the CALCULATE COLUMN command calculates. In this case, the calculation affects SALES. The ! is required after the member name.

The CALCULATE COLUMN command allows column numbers, row names, or constants in formulas. You can read the first calculation this way: "% equals column 1 as a percent of Sales in column 1."

Each calculated column label must be unique. Note how the second calculated column label has a blank space after the % sign.

To specify a constant, define a number followed by a period. You can use a constant in either a column or row calculation. The last column calculation takes the sum of columns 1 and 2 and divides by the value 2. This formula is interpreted as $(1+2)/2$, *not* $1 + (2/2)$.

As noted in [“Sample 7: Using Aliases” on page 1189](#), the ORDER command arranges columns in the specified order. By default, calculated columns are added to the end of existing columns retrieved from the database. In this example, columns 0-2 are automatically retrieved, based on selected members. Columns 3-5 are the calculated columns. The ORDER command applies to both retrieved and calculated columns.

This report script, COLCALC1.REP, is available in the `\ARBORPATH\App\Demo\Basic` directory, and is displayed in Enterprise View in Administration Services.

Sample 11: Placing Two-Page Layouts on the Same Page

This sample report has two different page layouts on the same page.

Year Profit_% Actual

	East	West	South	Market
Stereo	-0.52%	1.91%	0.00%	0.91%
Compact_Disc	32.60%	36.00%	0.00%	34.60%
Audio	17.86%	20.81%	0.00%	19.60%
Television	20.40%	16.57%	13.50%	17.21%
VCR	30.81%	32.43%	33.70%	32.24%
Camera	16.66%	21.66%	17.83%	19.07%
Visual	23.16%	23.56%	22.27%	23.09%
Product	21.34%	22.50%	22.27%	22.04%

Sales Actual Product

	Qtr1	Qtr2	Qtr3	Qtr4	Year
New_York	\$18,631	\$17,681	\$19,923	\$24,403	\$80,638
Boston	\$15,812	\$15,050	\$16,716	\$19,159	\$66,737
Chicago	\$16,536	\$15,599	\$17,411	\$21,374	\$70,920
East	\$50,979	\$48,330	\$54,050	\$64,936	\$218,295
San_Francisco	\$19,761	\$19,019	\$20,722	\$24,807	\$84,309
Seattle	\$13,766	\$13,546	\$14,204	\$19,034	\$60,550
Denver	\$13,800	\$13,588	\$13,838	\$18,232	\$59,458
Los_Angeles	\$17,866	\$17,269	\$17,208	\$22,635	\$74,978
West	\$65,193	\$63,422	\$65,972	\$84,708	\$279,295
Dallas	\$ 9,226	\$ 9,175	\$ 9,481	\$12,700	\$40,582
Houston	\$ 7,690	\$ 7,363	\$ 7,646	\$10,785	\$33,484
Phoenix	\$ 8,157	\$ 7,903	\$ 8,343	\$11,843	\$36,246
South	\$25,073	\$24,441	\$25,470	\$35,328	\$110,312
Market	\$141,245	\$136,193	\$145,492	\$184,972	\$607,902

Use the following script to create Sample 11:

```

<PAGE (Year, Accounts, Scenario)

    <COLUMN (Market)
    <ICHILDREN Market

<ROW(Product)
<IDESCENDANTS Product

Actual
{ DECIMAL 2 WIDTH 10 SUPBRACKETS AFTER "%" }
Profit_%
!

<PAGE (Accounts, Scenario, Product)
Actual
Sales
Product

    <COLUMN(Year)
    <ICHILDREN Year

<ROW(Market)

```

```
{ DECIMAL 0 After " " BEFORE "$" }
<IDESCENDANTS Market
!
```

In a single report, you can select multiple dimension layouts and members. To define a multiple layout report, define reports as you normally do. Separate the commands with exclamation marks as shown above. Whenever the column, row, or page dimensions change between ! output commands, new headings are automatically generated to match the new layout.

The BEFORE format command places a character in front of data values. The AFTER format command turns off the percent signs from the first report layout.

This report script, 2LAYOUTS.REP, is available in the \ARBORPATH\App\Demo\Basic directory, and is displayed in Enterprise View in Administration Services.

Sample 12: Formatting for Data Export

This sample creates a report with a member name in each column. This format is required when you export Essbase data to another product, such as an SQL database, with a flat file.

New York	Stereo	Sales	1000.0	950.0
New York	Stereo	Cost of Goods Sold	580.0	551.0
New York	Stereo	Margin	420.0	399.0
New York	Stereo	Marketing	80.0	80.0
New York	Stereo	Payroll	340.0	340.0
New York	Stereo	Miscellaneous	0.0	0.0
New York	Stereo	Total Expenses	420.0	420.0
New York	Stereo	Profit	0.0	-21.0
New York	Stereo	Profit %	0.0	-2.2
New York	Stereo	Margin %	42.0	42.0
New York	Compact Disc	Sales	1200.0	1150.0
New York	Compact Disc	Cost of Goods Sold	456.0	437.0
New York	Compact Disc	Margin	744.0	713.0
New York	Compact Disc	Marketing	95.0	95.0
New York	Compact Disc	Payroll	310.0	310.0
New York	Compact Disc	Miscellaneous	0.0	0.0
New York	Compact Disc	Total Expenses	405.0	405.0
New York	Compact Disc	Profit	339.0	308.0
New York	Compact Disc	Profit %	28.3	26.8
New York	Compact Disc	Margin %	62.0	62.0
New York	Audio	Sales	2200.0	2100.0
New York	Audio	Cost of Goods Sold	1036.0	988.0
New York	Audio	Margin	1164.0	1112.0
New York	Audio	Marketing	175.0	175.0
New York	Audio	Payroll	650.0	650.0
New York	Audio	Miscellaneous	0.0	0.0
New York	Audio	Total Expenses	825.0	825.0
New York	Audio	Profit	339.0	287.0
New York	Audio	Profit %	15.4	13.7
New York	Audio	Margin %	52.9	53.0
New York	Television	Sales	1800.0	1600.0

Use the following script to create Sample 12:

```
<PAGE (Scenario)
```

```

<COLUMN(Year)

<ROW (Market, Product, Accounts)
<CHILDREN East
<DESCENDANTS Product

{ DECIMAL 1
WIDTH 9
SUPBRACKETS
SUPCOMMA
MISSINGTEXT " "
UNDERSCORECHAR " "
SUPHEADING
NOINDENTGEN
SUPFEED
ROWREPEAT

Budget
    Jan Feb

<DESCENDANTS Accounts
    !

```

The ROWREPEAT command produces rows of data that have the member names repeated for each row dimension.

The SUPFEED command suppresses page feeds. A page feed automatically occurs when the report output reaches the default page length of 66 rows, unless you enter the PAGELENGTH command to change this setting. When a large flat file is created, you can use this command to prevent page breaks (blank rows) from being displayed in the report every time output reaches a logical page length.

This report script, `FLAT2SQL.REP`, is available in the `\ARBORPATH\App\Demo\Basic` directory, and is displayed in Enterprise View in Administration Services.

Sample 13: Creating Asymmetric Columns

Asymmetric columns make up this report. Typically, a report contains symmetric columns. That is, when multiple dimensions are displayed across the page as column groups, each level of nested columns has the same number of members nested below. Because Actual has only one nested column, Jan, and Budget has three nested columns, this report is considered asymmetric.

Some rows in the report use names other than the member names from the database. In addition to allowing aliases, as in [“Sample 7: Using Aliases” on page 1189](#), you can rename a row name in the reporter.

	Product Market			
	Actual	Budget	Budget	Budget
	Jan	Jan	Feb	Mar
	=====	=====	=====	=====
Revenue	49,896	49,950	45,770	45,770
Cost of Goods	20,827	19,755	18,058	18,047
Gross Margin	29,069	30,196	27,712	27,723

Marketing	3,560	3,515	3,525	3,515
Payroll	14,599	14,402	14,416	14,416
Miscellaneous	249	0	0	0
Total Expenses	18,408	17,917	17,941	17,931
Profit	10,661	12,279	9,771	9,792

Use the following script to create Sample 13:

```
<PAGE (Product, Market)

    <COLUMN (Scenario, Year)
        Actual   Budget Budget Budget
            Jan     Jan   Feb   Mar

<ROW (Accounts)

{ RENAME "Revenue" } Sales
{ RENAME "Cost of Goods" } Cost_of_Goods_Sold
{ RENAME "Gross Margin" } Margin

{ SKIP UNDERSCORECHAR " " }
<CHILDREN Total_Expenses

{ SKIP }
Profit
!
```

To create an asymmetric report, you must specify the member name of each column. Because the report output has two column groupings, Scenario and Year, you must specify a member from each dimension for each column. If you do not specify each column member, the resulting report format is symmetric.

The RENAME command redefines a member name when the report is output. Use the RENAME command when you do not want to use an alias table.

This report script, ASYMM.REP, is available in the \ARBORPATH\App\Demo\Basic directory, and is displayed in Enterprise View in Administration Services.

Sample 14: Calculating Columns

This section contains two examples of CALCULATE COLUMN scripts and the reports they produce. CALCULATE COLUMN supports standard mathematical operations.

- [“Sample 14-A: Basic Calculated Columns” on page 1200](#)
- [“Sample 14-B: Asymmetric Columns” on page 1201](#)

Sample 14-A: Basic Calculated Columns

East									
	Actual				Budget			Var	
Jan	Feb	Mar	Qtr1	Jan	Feb	Mar	Q1	Q1	

=====					=====				=====	
1,295	1,132	553	2,980	Tele~	Profit	1,240	950	950	3,140	(160)
25	27	14	66		Profit_%	26	22	22	70	(4)
56	62	59	177		Margin_%	60	60	60	180	(3)
1,417	1,120	898	3,435	VCR	Profit	1,466	1,161	1,161	3,788	(353)
33	30	24	87		Profit_%	35	31	31	98	(10)
61	61	62	183		Margin_%	63	63	63	189	(6)
400	272	256	928	Cam~	Profit	528	360	360	1,247	(319)
15	11	10	36		Profit_%	19	13	13	45	(10)
70	70	70	211		Margin_%	71	71	71	213	(2)
3,112	2,524	1,707	7,343	Visu~	Profit	3,234	2,471	2,471	8,175	(832)
25	24	17	66		Profit_%	27	23	23	74	(7)
61	63	63	187		Margin_%	64	64	64	191	(4)

Use the following script to create Sample 14-A:

```

<PAGE (Market)
East
  <COLUMN (Scenario, Year)
    Actual Budget
    Jan Feb Mar
{ CALCULATE COLUMN "Qtr1" = 2 : 4
  CALCULATE COLUMN "Q1" = 5 : 7
  CALCULATE COLUMN "Var~Q1" = 8 - 9

  ORDER 2,3,4,8,0,1,5,6,7,9
  WIDTH 7 WIDTH 10 0 1
}
<ROW (Product, Accounts)
<CHILDREN Visual

<CHILDREN Accounts
  !

```

This report script, COLCALC2.REP, is available in the \ARBORPATH\App\Demo\Basic directory, and is displayed in Enterprise View in Administration Services.

Sample 14-B: Asymmetric Columns

The following sample has two regular columns defined in *asymmetric* mode. For an explanation, including an example, of the use of asymmetric columns, see “[Sample 13: Creating Asymmetric Columns](#)” on page 1199.

East				
Budget		Actual		
Jan		Jan	% Sales	
=====		=====	=====	
1,200	Television	Payroll	1,236	25%
440		Marketing	365	9%
1,240		Profit	1,295	26%
4,800		Sales	5,244	100%
1,030	VCR	Payroll	1,044	25%
150		Marketing	156	4%

1,466		Profit	1,417	35%
4,200		Sales	4,311	100%
1,195	Camera	Payroll	1,167	42%
300		Marketing	288	11%
528		Profit	400	19%
2,850		Sales	2,656	100%
3,425	Visual	Payroll	3,447	29%
890		Marketing	809	8%
3,234		Profit	3,112	27%
11,850		Sales	12,211	100%

Use the following script to create Sample 14-B:

```
<PAGE(Market)
East

    <COLUMN(Scenario, Year)
    Budget Actual
    Jan     Jan

{ ORDER 2,0,1,3,4 WIDTH 12 0 1 NOINDENTGEN AFTER "%" 4
  SKIPONDIMENSION Product LMARGIN 10 }

<ROW(Product, Accounts)

{ CALCULATE ROW "Sales" OFF }
{ CALCULATE COLUMN "Actual~% Sales" = 2 % "Sales" 2 }

<ICHILDREN Visual
{ SAVEROW } Sales
    Payroll
    Marketing
    Profit
<DUPLICATE Sales
!
```

This report script, COLCALC3.REP, is available in the \ARBORPATH\App\Demo\Basic directory, and is displayed in Enterprise View in Administration Services.

Sample 15: Calculating Rows

The sample reports in this section demonstrate CALCULATE ROW scripts and the reports they produce.

- [“Sample 15-A: Basic Calculated Row” on page 1202](#)
- [“Sample 15-B: Calculated Rows and Missing Relationships” on page 1203](#)
- [“Sample 15-C: Rows of Averages” on page 1204](#)

Sample 15-A: Basic Calculated Row

This sample report demonstrates the basic form of the CALCULATE ROW command.

	Audio Actual Sales		
	Jan	Feb	Mar
	=====	=====	=====
Boston	1,985	1,801	1,954
New_York	2,310	2,082	2,259
Chicago	2,043	1,884	1,814
Total Sales	6,338	5,767	6,027
Avg Sales	2,113	1,922	2,009

Use the following script to create Sample 15-A:

```

Audio Actual Sales
Jan Feb Mar

{ CALCULATE ROW "Total Sales" } //create new calculated row
Boston
New_York
Chicago

{ SKIP
  CALCULATE ROW "Avg Sales" = "Total Sales" /3
  PRINTRROW "Total Sales"
  PRINTRROW "Avg Sales" }
!
```

This report script, ROWCALC1.REP, is available in the \ARBORPATH\App\Demo\Basic directory, and is displayed in Enterprise View in Administration Services.

Sample 15-B: Calculated Rows and Missing Relationships

This sample report is a simple summary of information in a North/South grouping, which is not part of the database outline. When relationships that you need for reporting are missing in the database outline, often the best solution is to use calculated rows (or columns).

	Budget Payroll		
	Jan	Feb	Mar
	====	====	====
Northern Cities			
=====			
New_York	1,940	1,930	1,930
Boston	1,610	1,610	1,610
Chicago	1,630	1,630	1,630
San_Francisco	1,815	1,815	1,815
Seattle	1,415	1,409	1,409
Southern Cities			
=====			
Denver	1,499	1,499	1,499
Los_Angeles	1,757	1,787	1,787
Dallas	1,002	1,002	1,002
Phoenix	900	900	900
Houston	834	834	834

Total Northern	8,410	8,394	8,394
Total Southern	5,992	6,022	6,022

Use the following script to create Sample 15-B:

```
// Declare Calculated Rows to Sum Southern and Northern Cities
{ CALCULATE ROW "Total Southern" OFF

// initially, set operation to OFF
  CALCULATE ROW "Total Northern" OFF  }

<PAGE(Product,Scenario,Accounts)
{ RENAME "" } Product           // all products, so blank out
                                // the Product Label
Budget
Payroll
  <COLUMN(Year)
    Jan Feb Mar

<ROW(Market)                   // Northern Cities

{ SETROWOP "Total Northern" +   // Accumulate for Northern

SKIP 3
IMMHEADING                      // Put out heading now so text
                                // will go after it
Text 0 "Northern Cities" UCHARACTERS
}

New_York Boston Chicago San_Francisco Seattle

//Southern Cities

{ SETROWOP "Total Southern" +   } // Accumulate for Southern
{ SETROWOP "Total Northern" OFF } // Stop Accumulation for Northern

{ SKIP Text 0 "Southern Cities" UCHARACTERS }

Denver Los_Angeles Dallas Phoenix Houston

{ SKIP
PRINTROW "Total Northern"       // output calculated rows
PRINTROW "Total Southern"
}
!
```

This report script, ROWCALC2.REP, is available in the \ARBORPATH\App\Demo\Basic directory, and is displayed in Enterprise View in Administration Services.

Sample 15-C: Rows of Averages

This report sample restricts columns during calculation to average rows that contain partly numbers and percentages. The report must calculate the total regional average percentages using previously calculated rows that contain the total sales for the region. Also, the report must compute (for averaging) a count of regions. The number of regions is set as a constant in the

database outline. If this number changes, the report definition must be modified. If a count of regions is not computed, a hard-to-notice error can result.

Actual Total Sales for the 3 Video Products in Qtr1: 36,914 35,126 25,119
 Budget Total Sales for the 3 Video Products in Qtr1: 37,300 34,250 26,940

=====

Qtr1

		Television		VCR		Camera	
		Profit	Profit_%	Profit	Profit_%	Profit	Profit_%
		=====	=====	=====	=====	=====	=====
New_York	Budget	1,020	20.40%	1,382	31.41%	540	16.68%
	Actual	847	17.66%	1,243	29.62%	352	11.79%
Boston	Budget	1,020	24.88%	1,344	35.37%	277	11.79%
	Actual	1,405	33.48%	1,002	27.49%	207	9.28%
Chicago	Budget	1,100	25.58%	1,062	31.24%	430	16.54%
	Actual	728	16.51%	1,190	30.68%	369	14.72%
San_Fran~	Budget	930	21.63%	718	21.12%	1,270	31.75%
	Actual	674	15.54%	1,197	31.12%	1,000	27.4%
Seattle	Budget	390	15.60%	973	32.98%	376	16.00%
	Actual	340	12.20%	977	31.56%	312	13.79%
Denver	Budget	690	22.26%	929	30.97%	462	18.86%
	Actual	334	11.94%	914	30.48%	361	15.92%
Los_Ange~	Budget	810	18.41%	1,101	29.76%	506	18.40%
	Actual	429	9.11%	1,127	28.81%	377	14.62%
Dallas	Budget	780	21.08%	1,341	36.24%	333	13.88%
	Actual	163	4.69%	1,055	30.28%	243	10.71%
Houston	Budget	690	24.64%	1,128	36.39%	432	18.00%
	Actual	256	10.44%	1,064	34.98%	241	10.98%
Phoenix	Budget	630	20.32%	894	31.93%	498	20.75%
	Actual	251	8.49%	940	31.07%	261	11.99%

Total Regions Averages

Avg	Budget	806	21.61%	1,087	31.74%	512	19.02%
Avg	Actual	543	14.70%	1,071	30.49%	372	14.82%

Use the following script to create Sample 15-C:

```
{ // Declare some of the Calculated Rows to be used
  CALCULATE ROW "Avg~Budget" OFF
  CALCULATE ROW "Avg~Actual" OFF
  CALCULATE ROW "Tot Sales~Budget" OFF
  CALCULATE ROW "Tot Sales~Actual" OFF
}
// We need the values of Market->Visual->Qtr1->Sales->Actual and
// Market ->Visual->Qtr1->Sales ->Budget to compute some
// percentages at the bottom, so get them now

Market
<CHILDREN Visual Qtr1 Sales
{ SAVEROW "Actual Sales" } Actual // stores into first 3
// data columns
{ SAVEROW "Budget Sales" } Budget // of
these rows, which
// are cols 1-3
// change to columns 2-4 when we
// specify 2 row dimensions in
```

```

// next section
// Since this is an example, not a formal report, we'll
// type out the values for Actual Sales and Budget Sales here so
// you can check the numbers:

{ SKIP 2
TEXT 0 "Actual Total Sales for the 3 Video Products in Qtr1:" 55 "*CALC" "Actual Sales"
TEXT 0 "Budget Total Sales for the 3 Video Products in Qtr1:" 55 "*CALC" "Budget Sales"
UCHARACTERS
SKIP 5 }

! // Now we can do the main report
{ AFTER "%" 3,5,7 DECI 2 3,5,7 ZEROTEXT "--" MISSING "--"
  WIDTH 10 0 1 }

<PAGE(Year)
Qtr1

    <COLUMN(Product,Accounts)
    <CHILDREN Visual
    Profit // split these 2 accounts onto
           // 2 lines to prevent default
    Profit_% // to asymmetric mode
            // because both column
            // dimensions have the same # of
            // members selected. Could have
            // used <SYM instead.

<ROW(Market,Scenario)
<ONSAMELEVELAS New_York
    { SETROWOP "Avg~Actual" OFF
      SETROWOP "Avg~Budget" +

      CALCULATE ROW "Count" = "Count" + 1. }

    Budget

    { SETROWOP "Avg~Budget" OFF
      SETROWOP "Avg~Actual" + }

    >{ SKIP }

    Actual

{ UCOLUMNS SKIP 2 }
{
// at this point, Avg~Budget and Avg~Actual ARE NOT YET
// AVERAGES--they are the SUM of the Profit rows of each type.
// Before converting them to averages, the report computes
// Profit as a % of total sales for each type. Since we only
// have 1 value for "Budget Sales" and "Actual Sales",
// for each of the three visual products in those
// rows, the report restricts the reference to those rows to
// columns 2-4 while computing the percentage columns 3, 5, and 7,
// based on profits in columns 2, 4 and 6
// calculate the percentages for Budget
CALCULATE ROW "Avg~Budget" 3 = "Avg~Budget" 2 % "Budget Sales" 2
CALCULATE ROW "Avg~Budget" 5 = "Avg~Budget" 4 % "Budget Sales" 3
CALCULATE ROW "Avg~Budget" 7 = "Avg~Budget" 6 % "Budget Sales" 4

```

```

// now calculate the averages
CALCULATE ROW "Avg~Budget" 2 = "Avg~Budget" / "Count"
CALCULATE ROW "Avg~Budget" 4 = "Avg~Budget" / "Count"
CALCULATE ROW "Avg~Budget" 6 = "Avg~Budget" / "Count"

// calculate the percentages for Actual
CALCULATE ROW "Avg~Actual" 3 = "Avg~Actual" 2 % "Actual Sales" 2
CALCULATE ROW "Avg~Actual" 5 = "Avg~Actual" 4 % "Actual Sales" 3
CALCULATE ROW "Avg~Actual" 7 = "Avg~Actual" 6 % "Actual Sales" 4

// now calculate the averages
CALCULATE ROW "Avg~Actual" 2 = "Avg~Actual" / "Count"
CALCULATE ROW "Avg~Actual" 4 = "Avg~Actual" / "Count"
CALCULATE ROW "Avg~Actual" 6 = "Avg~Actual" / "Count"

TEXT C "Total Regions Averages"
PRINTROW "Avg~Budget"
PRINTROW "Avg~Actual" }
!
```

This report script, ROWAVG.REP, is available in the \ARBORPATH\App\Demo\Basic directory, and is displayed in Enterprise View in Administration Services.

Sample 16: Sorting by Top or Bottom Data Values

The following two reports demonstrate the use of TOP and BOTTOM conditional retrieval commands in a report script. For a discussion of various issues related to use of the TOP and BOTTOM commands, see "Restricting and Ordering Data Values" in the *Oracle Essbase Database Administrator's Guide*.

- [“Sample 16-A: Bottom Data Values” on page 1207](#)
- [“Sample 16-B: Top Data Values” on page 1208](#)

Sample 16-A: Bottom Data Values

This sample report demonstrates the basic use of the BOTTOM command. The report is based on the Sample Basic database.

		Measures			
		Actual		Budget	
		Jan	Dec	Jan	Dec
		=====	=====	=====	=====
East	200	158	233	280	340
	300	184	277	240	210
	Diet	181	213	200	240
West	100	378	223	830	530
	300	755	971	830	950
	400	454	434	470	370
South	200	480	496	520	390
	Diet	355	404	490	430
	300	188	213	270	240
Central	300	790	824	930	810
	100	724	792	900	890

	400	691	785	660	650
Market	200	2,141	2,302	2,710	2,810
	300	1,917	2,285	2,270	2,210
	400	1,611	1,720	1,730	1,600

Use the following script to create Sample 16-A:

```
<Sym
<Column (Scenario, Year)
Actual Budget
Jan Dec
<Row (Market, Product)
<CHILDREN Market
<CHILDREN Product
<Bottom (3, @DataColumn(3))
!
```

The BOTTOM command specifies that only the three lowest data values are returned for each row grouping, based on the target data values specified in column three (Budget, Jan). Notice that no row dimension is selected here, so the report output defaults to the innermost row.

This report script, BOTTOM.REP, is available in the \ARBORPATH\App\Sample\Basic directory, and is displayed in Enterprise View in Administration Services.

Sample 16-B: Top Data Values

This sample report fragment demonstrates the basic use of the TOP command. The report is based on the Sample Basic database.

		Measures			
		Actual		Budget	
		Jan	Dec	Jan	Dec
		=====	=====	=====	=====
New York	100-10	262	271	260	250
	200-40	175	312	200	320
	400-10	101	89	120	90
	400-20	94	133	110	150
	300-10	111	309	100	210
	400-30	54	52	70	60
	300-20	(113)	(189)	(70)	(150)
	200-10	(172)	(224)	(170)	(210)
Massachusetts	100-10	367	390	360	360
	200-40	100	87	110	80
	400-10	29	29	40	40
	400-30	29	25	40	30
	300-10	17	7	30	10
	200-10	(23)	(20)	(10)	(10)
...					
East	100-10	837	867	860	830
	200-40	267	383	310	400
	400-10	215	201	280	230
	400-30	157	167	210	200
	300-10	177	368	190	270
	400-20	94	133	110	150
	200-20	80	79	100	110

100-20	67	122	70	110
100-30	20	37	30	50
300-30	34	12	30	20

Use the report script TOP.REP, reproduced here, to create Sample 16-B:

```
<Sym
//Suppress shared members from displaying
<Supshare
  <Column (Scenario, Year)
  Actual Budget
  Jan Dec
<Row (Market, Product)
<Desc Market
//Use bottom level of products
<DimBottom Product
<Top (10, @DataColumn(3))
!
```

The TOP command specifies that only the ten highest data values are returned for each row grouping, based on the target data values specified in column three (Budget, Jan).

This report script, TOP.REP, is available in the \ARBORPATH\App\Sample\Basic directory, and is displayed in Enterprise View in Administration Services.

Sample 17: Restricting Rows

The following report demonstrates the use of the RESTRICT conditional retrieval command in a report script. For a discussion of various issues related to use of the RESTRICT command, see "Restricting and Ordering Data Values" in the *Oracle Essbase Database Administrator's Guide*.

		Measures			
		Actual		Budget	
		Jan	Dec	Jan	Dec
		=====	=====	=====	=====
East	200	158	233	280	340
	300	184	277	240	210
	Diet	181	213	200	240
South	300	188	213	270	240
	400	#Missing	#Missing	#Missing	#Missing

Use the following script to create Sample 17:

```
<Sym
<Column (Scenario, Year)
Actual Budget
Jan Dec
<Row (Market, Product)
<Ichildren Market
<Ichildren Product
<Restrict (@DataCol(3) < $300.00 )
!
```

The RESTRICT command specifies that only data values that are less than \$300.00 are returned for each row grouping, based on the target data values specified in column three (Budget, Jan). Notice that no row dimension is selected here, so the report output defaults to the innermost row.

This report script, RESTRICT.REP, is available in the \ARBORPATH\App\Sample\Basic directory, and is displayed in Enterprise View in Administration Services.

Sample 18: Ordering Data Values

The following report demonstrates the use of the ORDERBY conditional retrieval command in a report script. For a discussion of various issues related to use of the ORDERBY command, see "Restricting and Ordering Data Values" in the *Oracle Essbase Database Administrator's Guide*.

		Sales Scenario			
		Jan	Feb	Mar	Apr
		=====	=====	=====	=====
New York	100-20	#Missing	#Missing	#Missing	#Missing
	100-30	#Missing	#Missing	#Missing	#Missing
	200-20	#Missing	#Missing	#Missing	#Missing
	200-30	#Missing	#Missing	#Missing	#Missing
	300-30	#Missing	#Missing	#Missing	#Missing
	Diet	#Missing	#Missing	#Missing	#Missing
	200-10	61	61	63	66
	400-30	134	189	198	198
	300-20	180	180	182	189
	400-20	219	243	213	223
	400-10	234	232	234	245
	300-10	483	495	513	638
	200-40	490	580	523	564
	200	551	641	586	630
	400	587	664	645	666
	300	663	675	695	827
	100-10	678	645	675	712
	100	678	645	675	712
	Product	2,479	2,625	2,601	2,835

Use the following script to create Sample 18:

```
<Page ("Measures")
<Column ("Scenario", "Year")
<Row ("Market", "Product")
"Sales"
"Scenario"
"Jan" "Feb" "Mar" "Apr"
"New York"
"Product" "100" "100-10" "100-20" "100-30" "200" "200-10"
"200-20" "200-30" "200-40" "300" "300-10" "300-20" "300-30" "400"
"400-10" "400-20" "400-30" "Diet" "100-20" "200-20" "300-30"

<ORDERBY ("Product", @DATACOL(1) ASC, @DATACOL(2) DESC, @DATACOL(3) ASC @DataCol (4)
DESC)
!
```

The ORDERBY command is based only on data in the data columns. If the SUPPRESSMISSING command is not used in the report, #MISSING is considered to be the lowest data value. ORDERBY compares data values in the following order:

- Two values in the same column (for example, in COL1, the value associated with 200-10 is compared with the 400-30 data value, as shown in the example below).
- Data values between two data columns (for example, the data value in COL1 is compared with the data value in COL2, as shown in the example next).

If two data values are the same, the sort proceeds to the next column to determine the order.

In the following subset of Sample 18, for Product 200-10, the data values in COL1 and COL2 are both 61; the data in COL1 should be in ascending order, the data in COL2 should be in descending order. The two values are compared, and as they are the same, COL2 and COL3 are compared. Therefore, even though COL2 is supposed to be in descending order, the comparison for the row 400-30 was determined by the values in COL3, which is in ascending order.

	COL 1	COL 2	COL 3	COL 4
	=====	=====		
200-10	61	61	63	66
400-30	134	189	198	198
300-20	180	180	182	189

The report script for Sample 18, ORDERBY.REP, is available in the \ARBORPATH\App\Sample\Basic directory, and is displayed in Enterprise View in Administration Services.

Sample 19: Narrowing Member Selection Criteria

The following report demonstrates the use of the LINK command to narrow the members returned in a selection in a report script. For a examples of use of the LINK command, see "Selecting Members by Using Boolean Operators" in the *Oracle Essbase Database Administrator's Guide*.

Market Measures Scenario		
	Qtr1	Qtr2
	=====	=====
100-10	5,096	5,892
100-20	1,359	1,534
100-30	593	446
200-10	1,697	1,734
200-20	2,963	3,079
200-30	1,153	1,231
200-40	908	986
300-10	2,544	3,231
300-20	690	815
300-30	2,695	2,723
400-10	2,838	2,998
400-20	2,283	2,522
400-30	(116)	(84)
100-20	1,359	1,534

200-20	2,963	3,079
300-30	2,695	2,723
Product	24,703	27,107

Use the following script to create Sample 19:

```
<Page (Market)
<Column (Year)
Qtr1 Qtr2
<Row (Product)
<Link (<UDA (product, naturally-flavored) OR <LEV (product, 0))
!
```

The LINK command uses the AND, OR, and NOT Boolean operators to refine the search. In the preceding example, the product with the "naturally-flavored" user-defined attribute (UDA), as well as all Level 0 products, are returned in the search.

Be careful how you group operators in the LINK expression. Essbase evaluates operators from left to right. Use parentheses to group the expressions. For example, A OR B AND C is the same as ((A OR B) AND C). In the first expression, Essbase evaluates the expression from left to right, evaluating A OR B before evaluating AND C. In the second expression, Essbase evaluates the subexpression in parentheses (A OR B) before the whole expression, producing the same result. However, if you use (A OR (B AND C)), Essbase evaluates the subexpression in parentheses (B AND C) before the whole expression, producing a different result.

This report script, LINK.REP, is available in the \ARBORPATH\App\Sample\Basic directory, and is displayed in Enterprise View in Administration Services.

Sample 20: Using Attributes in Member Selection

This sample report uses members of attribute dimensions to view data on base dimensions that are associated with those attribute dimensions.

Profit Actual Caffeinated_True Qtr1 East

	Ounces_32	Ounces_20	Ounces_16	Ounces_12	Ounces
	=====	=====	=====	=====	=====
Bottle	#Missing	488	240	(586)	142
Can	#Missing	#Missing	#Missing	2,776	2,776
Pkg Type	#Missing	488	240	2,190	2,918

Use the following script to create Sample 20:

```
{WIDTH 12}
<Page (Measures, Scenario, Caffeinated, Year, Market)
Profit
Actual
Caffeinated_True
Qtr1
East
<Column (Ounces)
<CHILDREN Ounces
<Row ("Pkg Type")
<CHILDREN "Pkg Type"
!
```

The report output reflects data on Quarter 1 profits for caffeinated products by all their available sizes and package types. The data values indicate #MISSING when there is no data for a specific size in a specific package type. Because attributes are defined only on sparse dimensions, there are several #MISSING values in the sample report. You can represent missing values by suppressing the row or substituting a replacement text string, such as N/A. See [“Sample 2: Handling Missing Values” on page 1179](#) for an example of substituting page breaks and labels for missing values.

This report script, ATTR.REP, is available in the \ARBORPATH\App\Sample\Basic directory, and is displayed in Enterprise View in Administration Services.

Sample 21: Using the WITHATTR Command in Member Selection

This sample report uses the WITHATTR command to view information based on the attributes of the members of a base dimension.

	Profit Actual Qtr1 East		
	Bottle	Can	Pkg Type
	=====	=====	=====
100-30	74	#Missing	74
200-30	#Missing	#Missing	#Missing
200-40	908	#Missing	908
400-10	645	#Missing	645
400-20	290	#Missing	290
400-30	545	#Missing	545

Use the following script to create Sample 21:

```
{WIDTH 12}
<Page (Measures, Scenario, Year, Market)
Profit
Actual
Qtr1
East
<Column ("Pkg Type")
<CHILDREN "Pkg Type"
<Row (Product)
<WITHATTR (Caffeinated, "<>", True)
<IDESCENDANTS Product
!
```

The report output reflects data on Quarter 1 profits for caffeinated products by their package types. The data values indicate #MISSING when there is no data for a specific package type. Because attributes are defined only on sparse dimensions, there are several #MISSING values in the sample report.

This report script, WITHATTR.REP, is available in the \ARBORPATH\App\Sample\Basic directory, and is displayed in Enterprise View in Administration Services.

Report Writer Command Reference

Consult the Contents pane for a categorical list of Report Writer commands.

&	LEV	SAVEROW
!	LINK	SCALE
ACCOFF	LMARGIN	SETCENTER
ACCON	MASK	SETROWOP
AFTER	MATCH	SINGLECOLUMN
ALLINSAMEDIM	MATCHEX	SKIP
ALLSIBLINGS	MEANINGLESTEXT	SKIPONDIMENSION
ANCESTORS	MISSINGTEXT	SORTALTNAMES
ASYM	NAMESCOL	SORTASC
ATTRIBUTE	NAMESON	SORTDESC
ATTRIBUTEVA	NAMEWIDTH	SORTGEN
BEFORE	NEWPAGE	SORTLEVEL
BLOCKHEADERS	NOINDENTGEN	SORTMBRNAMES
BOTTOM	NOPAGEONDIMENSION	SORTNONE
BRACKETS	NOROWREPEAT	SPARSE
CALCULATE COLUMN	NOSKIPONDIMENSION	STARTHEADING
CALCULATE ROW	NOUNAMEONDIM	SUDA
CHILDREN	OFFCOLCALCS	SUPALL
CLEARALLROWCALC	OFFROWCALCS	SUPBRACKETS
CLEARROWCALC	OFSAMEGEN	SUPCOLHEADING
COLHEADING	ONCOLCALCS	SUPCOMMAS
COLUMN	ONROWCALCS	SUPCURHEADING
COMMAS	ONSAMELEVELAS	SUPEMPTYROWS
CURHEADING	ORDER	SUPEUROPEAN
CURRENCY	ORDERBY	SUPFEED
DATEFORMAT	OUTALT	SUPFORMATS
DECIMAL	OUTALTMBR	SUPHEADING

&	LEV	SAVEROW
DESCENDANTS	OUTALT/NAMES	SUPMASK
DIMBOTTOM	OUTALT/SELECT	SUPMISSINGROWS
DIMEND	OUTFORMATTEDMISSING	SUPNAMES
DIMTOP	OUTFORMATTEDVALUES	SUPOUTPUT
DUPLICATE	OUTMBRALT	SUPPAGEHEADING
ENDHEADING	OUTMBR/NAMES	SUPSHARE
EUROPEAN	OUTMEANINGLESS	SUPSHAREOFF
FEEDON	OUTPUT	SUPZEROROWS
FIXCOLUMNS	OUTPUTMEMBERKEY	SYM
FORMATCOLUMNS	PAGE	TABDELIMIT
GEN	PAGEHEADING	TEXT
HYBRIDANALYSIS/ON	PAGELENGTH	TODATE
HYBRIDANALYSIS/OFF	PAGEONDIMENSION	TOP
HEADING	PARENT	UCHARACTERS
IANCESTORS	PERSPECTIVE	UCOLUMNS
ICHILDREN	PRINTROW	UDA
IDESCENDANTS	PYRAMIDHEADERS	UDATA
IMMHEADING	QUOTEMBR/NAMES	UNAME
INCEMPTYROWS	REMOVECOLCALCS	UNAMEONDIMENSION
INCFORMATS	RENAME	UNDERLINECHAR
INCMASK	REPALIAS	UNDERSCORECHAR
INCMISSINGROWS	REPALIASMBR	WIDTH
INCZEROROWS	REPMBR	WITHATTR
INDENT	REPMBRALIAS	WITHATTREX
INDENTGEN	REPQUALMBR	ZEROTEXT
IPARENT	RESTRICT	
LATEST	ROWREPEAT	
LEAVES	SAVEANDOUTPUT	

&

Prefaces a substitution variable in the report script.

Syntax

& variableName

Parameter	Description
-----------	-------------

variableName	The name of the substitution variable set on the database.
--------------	------------------------------------------------------------

Notes

Any string that begins with a leading & is treated as a substitution variable; Essbase replaces these variables with their associated values prior to the parsing of the report script. Member names beginning with & are considered substitution variables by Report Writer.

Example

```
<ICHILDREN &CurQtr
```

becomes

```
<ICHILDREN Qtr1
```

if the substitution variable CurQtr has the value name "Qtr1".

See Also

- [& in calculation scripts](#)

!

Tells Essbase to output the instructions in the report script to the current line.

Syntax

!

Notes

Each report script requires at least one ! command to produce output. Use multiple instances of the ! command to separate multiple report specifications in a report script.

Following !, the new report specification retains data format output commands from previous specifications unless you enter commands in the new report that turn them off. The new report specification does not retain data extraction command defaults.

If you omit ! at the end of the report script and run the report, the report processor does not report output or display an error message.

ACCOFF

Turns off member accumulation.

Note: By default, the report script uses <ACCOFF.

Syntax

<ACCOFF

Notes

<ACCOFF selects members of the same dimension only if the select commands of the dimension follow one another in the report script. If a select command containing another dimension interrupts, the report script ignores the previous select commands. <ACCOFF can be used in multiple report scripts where the script redefines only a few select statements from the previous script.

Example

In the following report script, <ACCOFF excludes the two members that precede East (100-10 and 200-10), because East is from a different dimension. The report script includes 300-10 and 400-10, which follow East.

```
<PAGE (Measures)
Sales
<ASYM
<COLUMN (Scenario, Year)
Actual Budget
Jan Feb
<ROW (Product, Market)
<ACCOFF
"100-10 "
"200-10 "
"East "
"300-10 "
"400-10 "
!
```

This example produces the following report:

		Sales	
		Actual	Budget
		Jan	Feb
		=====	=====
300-10	East	999	770
400-10	East	562	580

See Also

- [ACCON](#)

ACCON

Turns on member accumulation.

Note: By default, member accumulation is off.

Syntax

<ACCON

Notes

This command selects all members, regardless of the order of the select statements. Use this command to mix members from different dimensions in select statements.

Example

In the following report script, the <ACCON command includes all members in the report script, regardless of dimensionality.

```
<PAGE (Measures)
Sales
<ASYM
<COLUMN (Scenario, Year)
Actual Budget
Jan Feb
<ROW (Product, Market)
<ACCON
"100-10 "
"200-10 "
"East "
"300-10 "
"400-10 "
!
```

This example produces the following report:

		Sales	
		Actual	Budget
		Jan	Feb
		=====	=====
100-10	East	1,812	1,640
200-10	East	647	630
300-10	East	999	770
400-10	East	562	580

See Also

- [ACCOFF](#)

AFTER

Displays a character following the data columns in the report.

This command displays only the first character of a string, even if more are specified. If you do not specify any columns in *columnList*, *char* is displayed after all data columns in the report.

Syntax

```
{ AFTER char [columnList] }
```

Parameter Description

char	A single-byte character enclosed in quotation marks.
columnList	Optional list of one or more column numbers, separated by spaces. If included, AFTER affects only these columns. If you do not specify <i>columnList</i> , all data columns are affected.

Notes

- Double-byte characters are not supported.
- If a value is equal to #MISSING, the string inserted after it does not print, even if you replace #MISSING with some other value (such as 0).

Example

The {AFTER "%"} command in the following report displays the percent sign after each data value.

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual

<COLUMN (Year)
<CHILDREN Year

<ROW (Product)

{ AFTER "%" }
<CHILDREN Audio
!
```

This example produces the following report:

```
Chicago Sales Actual

      Qtr1   Qtr2   Qtr3   Qtr4   Year
=====  =====  =====  =====  =====
Stereo      2,591%  2,476%  2,567%  3,035%  10,669%
Compact_Disc 3,150%  3,021%  3,032%  3,974%  13,177%
  Audio      5,741%  5,497%  5,599%  7,009%  23,846%
```

See Also

- [BEFORE](#)

ALLINSAMEDIM

Selects all the members from the same dimension as the specified dimension member for the report.

Syntax

```
<ALLINSAMEDIM mbrName
```

Parameter Description

mbrName Single member representing a dimension. All members from this dimension are selected.

Notes

ALLINSAMEDIM is not supported with HYBRIDANALYSISON.

Example

```
<ALLINSAMEDIM Audio
```

Selects all the members from the dimension for the following report.

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual
```

```
<COLUMN (Year)
<CHILDREN Year
```

```
<ROW (Product)
<ALLINSAMEDIM Audio
!
```

This example produces the following report:

```
                Chicago Sales Actual
                Qtr1   Qtr2   Qtr3   Qtr4   Year
                =====
Stereo          2,591   2,476   2,567   3,035   10,669
Compact_Disc   3,150   3,021   3,032   3,974   13,177
  Audio        5,741   5,497   5,599   7,009   23,846
Television     4,410   4,001   4,934   6,261   19,606
VCR            3,879   3,579   4,276   4,877   16,611
Camera         2,506   2,522   2,602   3,227   10,857
  Visual      10,795  10,102  11,812  14,365  47,074
  Product     16,536  15,599  17,411  21,374  70,920
```

See Also

- [ALLSIBLINGS](#)
- [DESCENDANTS](#)

ALLSIBLINGS

Adds all the siblings of the specified member to the report.

Syntax

```
<ALLSIBLINGS mbrName
```

Parameter Description

mbrName Name of member whose siblings you want to add.

Example

```
<ALLSIBLINGS Stereo
```

selects the siblings of the member Stereo for the following report script:

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual
```

```
<COLUMN (Year)
<CHILDREN Year
```

```
<ROW Product)
<ALLSIBLINGS Stereo
!
```

This example produces the following report:

```
Chicago Sales Actual

      Qtr1   Qtr2   Qtr3   Qtr4   Year
=====  =====  =====  =====  =====
Stereo      2,591   2,476   2,567   3,035  10,669
Compact_Disc 3,150   3,021   3,032   3,974  13,177
```

See Also

- [ANCESTORS](#)
- [DESCENDANTS](#)

ANCESTORS

Adds all the ancestors of the specified member to the report.

Syntax

```
<ANCESTORS mbrName
```

Parameter Description

mbrName Name of member whose ancestors you want to add.

Example

```
<ANCESTORS Stereo
```

Adds Audio and Product to the following report since Audio is the parent to Stereo and Product is the parent to Audio.

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual
```

```
<COLUMN (Year)
<CHILDREN Year
```

```
<ROW (Product)
```

<ANCESTORS Stereo
!

This example produces the following report:

	Chicago Sales Actual				
	Qtr1	Qtr2	Qtr3	Qtr4	Year
	=====	=====	=====	=====	=====
Audio	5,741	5,497	5,599	7,009	23,846
Product	16,536	15,599	17,411	21,374	70,920

See Also

- [ANCESTORS](#)

ASYM

Causes a report to be printed in an asymmetric format.

This command reverses a previously specified SYM command in an asymmetric report.

If <SYM is used, all report headers appear in a symmetric format, even if there are equal numbers of members in each row of the column header. <ASYM turns off symmetric mode.

Note: Essbase prints an asymmetric report (with BLOCKHEADERS) only when all column dimensions include the same number of selected members and all members from each column dimension are on the same line. Otherwise, a symmetric report (with PYRAMIDHEADERS) is produced.

Syntax

<ASYM

Notes

If the number of members you select from one column dimension differs from the number of members you select from another column dimension, the resulting report is *always* symmetric.

Example

The following example is based on Sample Basic.

```
<PAGE (Measures, Market)
South Sales
<SYM
  <COLUMN (Scenario, Year)
    Actual Budget
    Jan Feb
<ROW (Product)
<IDESCENDANTS "100"
!
<ASYM
!
```

Which produces the following reports:

```
          Sales Texas
          Actual      Budget
          Jan      Feb      Jan      Feb
          =====
100-10          452      465      560      580
100-20          190      190      230      230
100-30      #Missing #Missing #Missing #Missing
   100          642      655      790      810
```

```
          Sales Texas
          Actual      Budget
          Jan      Feb
          =====
100-10          452      580
100-20          190      230
100-30      #Missing #Missing
   100          642      810
```

See Also

- [SYM](#)

ATTRIBUTE

Returns all base-dimension members associated with a specified attribute.

Syntax

```
<ATTRIBUTE attrMbrName
```

Parameter	Description
-----------	-------------

<i>attrMbrName</i>	The name of a member of an attribute dimension.
--------------------	-------------------------------------------------

Notes

- When *attrMbrName* is a non level-0 member of an attribute dimension, Essbase returns all base-dimension members associated with its children. For example, in the Sample Basic database, `<ATTRIBUTE Large` returns all base-dimension members associated with any children of the attribute parent Large.
- With Boolean attributes, if you specify a Boolean dimension name (for example, Caffeinated), Essbase returns all base-dimension members associated with either Caffeinated member (for example, True or False). To return only one or the other, specify that member name (for example, `<ATTRIBUTE Caffeinated_True`).
- Your outline may contain duplicate Boolean, date, and numeric attribute-dimension member names; for example, 12 can be the attribute value for the size (in ounces) of a product as well as the value for the number of packing units for a product. To distinguish duplicate

member names with the <ATTRIBUTE command, specify the full name of the attribute (for example, <ATTRIBUTE 12_Ounces).

Example

```
<ATTRIBUTE Red
```

returns all base-dimension members associated with the member Red of the specified attribute dimension.

```
<PAGE (Market, Measures, Scenario)
      South Sales Actual
```

```
<COLUMN (Year)
<CHILDREN Year
```

```
{OUTALTNames}
<ATTRIBUTE Ounces_12
```

!

returns on rows only the names of the drinks that are associated with the member Ounces_12 on the corresponding attribute dimension:

	South Sales Actual				
	Qtr1	Qtr2	Qtr3	Qtr4	Year
	=====	=====	=====	=====	=====
Cola	2,296	2,509	2,975	2,824	10,604
Diet Cola	1,436	1,569	1,482	1,189	5,676
Old Fashioned	1,686	1,625	1,773	1,840	6,924
Sasparilla	1,862	1,938	1,830	1,921	7,551
Diet Cream	1,241	1,255	1,378	1,593	5,467

See Also

- [WITHATTR](#)

ATTRIBUTEVA

Returns all base-dimension members associated with a specified varying attribute member. This command allows querying of the base member list given the attribute member-dimension and the perspective setting.

Note: For use only in applications enabled with varying attributes.

Syntax

```
<ATTRIBUTEVA (attMbrName, options, startTuple[, endTuple])
```


Parameter	Description
attrMbrName	The name of a member of a varying attribute dimension.
options	ANY
startTuple[, endTuple]	(m1, m2, . . . , mN) Level-0 members from one or more independent dimensions for attrMbrName may be part of the input tuple. Members from all independent dimensions should be listed. If a member is not listed, the member of the same dimension from the current query or calculation context is used.

Notes

- When *attrMbrName* is a non level-0 member of an attribute dimension, Essbase returns all base-dimension members associated with its children.
- With Boolean attributes, if you specify a Boolean dimension name (for example, Caffeinated), Essbase returns all base-dimension members associated with either Caffeinated member (for example, True or False). To return only one or the other, specify that member name (for example, <ATTRIBUTEVA Caffeinated_True).
- Your outline may contain duplicate Boolean, date, and numeric attribute-dimension member names; for example, 12 can be the attribute value for the size (in ounces) of a product as well as the value for the number of packing units for a product. To distinguish duplicate member names with the <ATTRIBUTEVA command, specify the full name of the attribute (for example, <ATTRIBUTE 12_Ounces).

Example

```
<AttributeVa([Ounces_12], ANY, (Jan), (Feb))
```

```
<AttributeVa([Ounces], ANY, (Jan))
```

See Also

- [WITHATTR](#)
- [PERSPECTIVE](#)

BEFORE

Displays a character string before data columns in the report.

Quotes without a character string clear the text displayed before data columns. For example, { BEFORE "" } turns off previously issued BEFORE commands.

Syntax

```
{ BEFORE "char" [ columnList ] }
```

Parameter Description

char	A single-byte character enclosed in quotation marks.
------	------------------------------------------------------

Parameter Description

`columnList` **Optional.** List of the column numbers, separated by spaces, that you want *char* to precede. Without *columnList*, *char* is displayed before all columns in the report.

Notes

- Double-byte characters are not supported.

Example

`{ BEFORE "$" }` displays the dollar sign before all the data values in the following report:

```
<PAGE Market, Accounts, Scenario)
Chicago Sales Actual
  <COLUMN Year)
  <CHILDREN Year
<ROW (Product)
{ BEFORE "$" }
<CHILDREN Audio
!
```

This example produces the following report:

	Chicago Sales Actual				
	Qtr1	Qtr2	Qtr3	Qtr4	Year
	=====	=====	=====	=====	=====
Stereo	\$2,591	\$2,476	\$2,567	\$3,035	\$10,669
Compact_Disc	\$3,150	\$3,021	\$3,032	\$3,974	\$13,177
Audio	\$5,741	\$5,497	\$5,599	\$7,009	\$23,846

See Also

- [AFTER](#)

BLOCKHEADERS

Displays all members that apply to a column as the column heading, in the style used by asymmetric reports.

Note: This is the only format that can be used with asymmetric reports. Pyramid headers are the default for symmetric reports.

Syntax

```
{ BLOCKHEADERS }
```

Notes

- BLOCKHEADERS is a setting command.
- BLOCKHEADERS can be useful when columns are reordered and previously symmetric upper-tier column headers no longer align properly.
- BLOCKHEADERS ensures right-justified alignment of all columns.

Example

The following example is based on Sample Basic.

```
<PAGE Measures)
Sales
{WIDTH 7}
{BLOCKHEADERS}
<SYM
    <COLUMN (Scenario, Year, Market)
    Actual Budget
    Jan Feb
    East West
<ROW (Market)
<IDESCENDANTS "400"
!
```

This example produces the following report:

	Sales							
	Actual	Actual	Actual	Actual	Budget	Budget	Budget	Budget
	Jan	Jan	Feb	Feb	Jan	Jan	Feb	Feb
	East	West	East	West	East	West	East	West
	=====	=====	=====	=====	=====	=====	=====	=====
400-10	562	1,115	560	1,122	580	740	580	740
400-20	219	1,032	243	1,065	230	690	260	700
400-30	432	625	469	618	440	410	490	400
400	1,213	2,772	1,272	2,805	1,250	1,840	1,330	1,840

See Also

- [PYRAMIDHEADERS](#)

BOTTOM

Returns rows with the lowest values of a specified data column.

Syntax

```
<BOTTOM ([rowgroupDimension,] rows, column)
```

Parameter	Description
rowgroupDimension	Optional row grouping dimension that determines the rows to sort as a set. Default value: inner row.
rows	Number of rows to be returned; must be greater than 0.
column	@DATACOL (<i>colnumber</i>) @DATACOL (<i>colnumber</i>) where <i>colnumber</i> is the target column number; must be between 1 and the maximum number of columns in the report.

Notes

This command sorts the result set by the value of the specified data column in descending order.

Rows containing #MISSING values in the sort column are discarded from the result set before BOTTOM is applied.

You can use TOP and BOTTOM, ORDERBY and RESTRICT in the same report script, but you can use each command only once per report. If you repeat the same command in a second report in the same report script, the second command overwrites the first. Place global script formatting commands before a PAGE, COLUMN command or associated member (for example, <ICHILDREN or <IDESCENDANTS). Avoid using row formatting commands with BOTTOM.

If any of the ORDERBY, TOP, BOTTOM, or RESTRICT commands exist together in a report script, *rowgroupDimension* should be the same. Otherwise, an error is issued.

The ORDERBY, TOP, and BOTTOM commands sort a report output by its data values. The RESTRICT command restricts the number of valid rows for the report output. Their order of execution is:

1. Any sorting command that sorts on member names (for example <SORTDESC or <SORTASC)
2. RESTRICT
3. TOP and BOTTOM
4. ORDERBY

This order of execution applies regardless of the order in which the commands appear in the report script.

You can use configurable settings to specify the size of the internal buffers used for storing and sorting the extracted data. The following settings affect the way the RESTRICT, TOP, and BOTTOM commands work:

- Retrieval Buffer Size (a database setting)
- Retrieval Sort Buffer Size (a database setting)
- “NUMERICPRECISION” on page 477 (an *essbase.cfg* setting)

For more information on the database settings, see the *Oracle Essbase Database Administrator's Guide*.

Example

Example 1:

```
<Page (Market, Accounts, Scenario)
  Chicago Sales Actual
<Bottom (5, @DataCol(4))
<Column(Year)
<Ichildren Year
<Row(Product)
<Idescendants Product
!
<Bottom (3, @DataCol(1))
{Indentgen 3}
Boston Sales Actual
<Ichildren Year
```

```
<Idescendants Product
!
```

Which produces the following report based on the Demo Basic sample database:

	Chicago Sales Actual				
	Qtr1	Qtr2	Qtr3	Qtr4	Year
	=====	=====	=====	=====	=====
Television	4,410	4,001	4,934	6,261	19,606
VCR	3,879	3,579	4,276	4,877	16,611
Compact_Disc	3,150	3,021	3,032	3,974	13,177
Camera	2,506	2,522	2,602	3,227	10,857
Stereo	2,591	2,476	2,567	3,035	10,669

```
-----
```

	Boston Sales Actual				
	Qtr1	Qtr2	Qtr3	Qtr4	Year
	=====	=====	=====	=====	=====
Compact_Disc	3,290	3,034	3,132	3,571	13,027
Stereo	2,450	2,341	2,377	2,917	10,085
Camera	2,230	2,255	2,266	3,162	9,913

	Boston Sales Actual				
	Qtr1	Qtr2	Qtr3	Qtr4	Year
	=====	=====	=====	=====	=====
Compact_Disc	3,290	3,034	3,132	3,571	13,027
Stereo	2,450	2,341	2,377	2,917	10,085
Camera	2,230	2,255	2,266	3,162	9,913

Example 2:

The following example uses the ORDERBY, TOP, BOTTOM, and RESTRICT functions:

```
<TOP ("Year", 10, @DataCol(2))
{Width 15}
{Decimal 2}
{OutAltNames}
<BOTTOM ("Year", 5, @DataCol(2))
<OutMBrAlt
<Column(Scenario)
  {SupBrackets}
  Actual Budget "Variance %"
<RESTRICT (@DataCol(2) > 3000 and @DataCol(1)
  < 3500)
<Row(Year, Product)
<Idescendants Product
<Children Year
<OrderBy ( "Year",@DataCol(1), @DataCol(2) Desc)
!
```

Which produces the following report based on the Sample Basic sample database:

	Measures Market			
		Actual	Budget	Variance %
		=====	=====	=====
Qtr2	100-20 Diet Cola	1,534.00	21,010.00	-38.15
	100-20 Diet Cola	1,534.00	21,010.00	-38.15

	300-30 Diet Cream	2,723.00	3,100.00	-12.16
	300-30 Diet Cream	2,723.00	3,100.00	-12.16
Qtr4	300-30 Diet Cream	2,820.00	3,080.00	-8.44
	300-30 Diet Cream	2,820.00	3,080.00	-8.44
	200-20 Diet Root	2,834.00	3,790.00	-25.22
	200-20 Diet Root	2,834.00	3,790.00	-25.22
Qtr1	200-20 Diet Root	2,963.00	3,600.00	-17.69
Qtr2	200-20 Diet Root	3,079.00	3,640.00	-15.41
	200-20 Diet Root	3,079.00	3,640.00	-15.41
Qtr3	200-20 Diet Root	3,149.00	3,700.00	-14.89
	200-20 Diet Root	3,149.00	3,700.00	-14.89
	400-10 Grape	3,201.00	3,090.00	3.59
	300-10 Dark Cream	3,355.00	3,730.00	-10.05

See Also

- [RESTRICT](#)
- [TOP](#)
- [ORDERBY](#)

BRACKETS

Displays parentheses around negative numbers instead of negative signs.

Note: Brackets are the default for negative numbers.

Syntax

```
{ BRACKETS }
```

Notes

The BRACKETS command need only be used to cancel the effect of a previously issued SUPBRACKETS command. Brackets are used by this command to mean parentheses.

Example

{BRACKETS} displays -43.243 as (43.243) in the report.

See Also

- [SUPBRACKETS](#)

CALCULATE COLUMN

Creates a new report column, performs on-the-fly calculations, and displays the calculation results in the newly-created column.

Each new calculated column is appended to the right of the existing columns in the order in which it is created, and is given the next available column number.

See [ORDER](#) for more information on column numbering and ordering.

Syntax

```
{ CALCULATE COLUMN "newColumn" = expression }
```

Parameter	Description
-----------	-------------

"newColumn"	New column name enclosed by quotation marks.
-------------	----------------------------------------------

expression	<p>A column calculation expression.</p> <p>If an operation or equation is not specified, the default is + (add).</p> <p>The following mathematical operators are supported in column calculations:</p> <ul style="list-style-type: none">+ Addition operator.- Subtraction operator.* Multiplication operator.%X%Y Evaluates X as a percentage of Y./ Division operator.:X:Y Performs a summation of data values from X to Y (inclusive). Must be the first operator if used with multiple operators.
------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Notes

- No more than 50 column calculations can be defined at any one time in the report.
- All arguments in expressions must be valid data column numbers, as determined by the original order of the columns, or constants. Floating point constants can be entered directly into an expression (for example 0.05). Integer values are designated by a decimal point following the last digit (for example, 10.); this distinguishes integer constants from column references. For example, the following command sums columns 1 through 12 and divides the total by 12:

```
{CALCULATE COLUMN "New_Col" = 1+3 / 6+8 % 15 * 100.-"Tot_Row" 3+12}
```

- Precede and follow all operators in an expression with a single space.
- Nested (parenthetical) expressions are not supported.
- Expressions are *always* evaluated left to right, regardless of operator precedence. For example, the expression 1 + 4 + 5 / 100.0 sums columns 1, 4, and 5, and divides the total by 100. To sum columns 1 and 4 and add the quotient of column 5 divided by 100, use the following expression: 5 / 100.0 + 1 + 4
- You can use the ORDER command to arrange columns in an easy-to-read fashion.
- If you use the same name for more than one column, Essbase creates only the last column specified in the CALC COLUMN command. Use a leading space with the second (or two leading spaces with the third, and so on) name to create a "unique" column name.
- The SUM RANGE operator (:) can only be used as the first operation in an expression. For example, = 1 : 3 or = 1 : 3 + 7 * 9 are valid expressions, but =7* 9 : 12 is invalid because the SUM RANGE operator is not the first operator. The SUM RANGE operator (:) may not be used with a calculated row as one of the arguments. For example, = 1 : "Total_Sales" 3 is invalid.

- A reference to a calculated row in a column calculation must include a column restriction to specify the single column whose value is to be used in the calculation.
- A column calculation cannot reference a calculated row name that has not yet been declared. Use { CALCULATE ROW "calcrowname" OFF } prior to the CALCULATE COLUMN referencing it, to declare a calculated row's name when the actual definition of the row calculation's operation cannot be done until later in the report.
- If a column calculation is attached to a member that is nested within a repeating group, it is redefined over and over. This is allowed, but very inefficient. When possible, define column calculations prior to areas of the report where members repeat. If the same name occurs later in the report with a new and different definition, the prior definition is lost.

Example

Example 1 (CALCULATE COLUMN)

The following example is based on Sample Basic.

```
<PAGE (Measures, Market)
Sales
<SYM
  <COLUMN (Scenario, Year)
    Actual Budget
    Jan Feb
{WIDTH 8 0}
{WIDTH 7}
{WIDTH 11 5 6}
{CALCULATE COLUMN "Actual YTD" = 1 + 2}
{CALCULATE COLUMN "Budget YTD" = 3 + 4}
{ORDER 0 1 2 5 3 4 6}
<ROW (Market)
<CHILD "400"
  !
```

This example produces the following report:

	Sales		Market		Budget			
	Actual		Actual	Ytd	Jan	Feb	Budget	Ytd
	Jan	Feb	Actual	Ytd	Jan	Feb	Budget	Ytd
	=====	=====	=====	=====	=====	=====	=====	=====
400-10	2,839	2,879	5,718	2,320	2,350		4,670	
400-20	2,562	2,596	5,158	2,040	2,050		4,090	
400-30	1,233	1,261	2,494	990	1,030		2,020	

Example 2 (CALCULATE COLUMN)

The following samples demonstrate additional column calculations.

To calculate a new column named "1st Qtr" equal to the sum of the first 3 columns:

```
{CALCULATE COLUMN "1st Qtr" = 1 : 3}
```

To calculate a new column that is equal to column 12 taken as a percentage of the value in column 12 of a calculated row called "Total Sales":

```
{CALCULATE COLUMN "% of Total" = 12 % "Total Sales" 12}
```


To calculate a new column equal to column 1 multiplied by the constant 35:

```
{CALCULATE COLUMN "Extended_Price" = 1 * 35.}
```

The following example calculates a new column, adds column 1 to column 3, divides the result by column 6, adds column 8, takes that result as a percentage of column 15, multiplies that result by the constant number 100, subtracts the value from the 3rd column of the calculated row "Tot_Row", and adds the result to column 12.

```
{CALCULATE COLUMN "New_Col" = 1+3 / 6+8 % 15 * 100.-"Tot_Row" 3+12}
```

See Also

- [OFFCOLCALCS](#)
- [ONCOLCALCS](#)
- [REMOVECOLCALCS](#)
- [SETROWOP](#)

CALCULATE ROW

Creates a named row and associates it with a row name or label. This is similar to declaring a variable. This command can also specify an operation (+, -, *, /, or OFF) as an equation consisting of constants, other calculated rows, and operators.

Equations are evaluated at the time of declaration. If an operator is specified, subsequent output rows have the operator applied to them with the result stored in the calculated row.

This is useful for aggregating a series of rows to obtain a subtotal or total. The operator can be reset at any point with SETROWOP. If neither an equation nor an operator are specified in the CALCULATE ROW command, the + operator is assumed.

SETROWOP defines a calculation operator to be applied to all subsequent output data rows. Use PRINTROW to display the calculation results in the newly created row.

Syntax

1:

```
{ CALCULATE ROW "newRow" [ columnNo ] = expression }
```

2:

```
{ CALCULATE ROW "newRow" [ operator ] }
```

Parameter Description

"newRow" Name of a new row, enclosed by quotation marks, that was declared with SAVEROW or SAVEANDOUTPUT.

columnNo **Optional.** Column numbers to which Esbase applies the expression.

expression Row calculation expression. Member names are not supported.

Parameter Description

operator One of the following mathematical operators:

- + Addition.
- – Subtraction.
- * Multiplication.
- %X%Y X as a percentage of Y.
- / Division.
- OFF Turns off the row operator.

If omitted, the default is + (add).

Notes

- Row name can have multiple levels, separated by the tilde (~) character, for use when there is more than one row name column in the report. For example, the calculated row name "Actual~Sales", if output (using PRINTROW) in a report with at least two row name columns, results in Sales in the right-most row name column, and Actual in the row name column to its left. If a multiple level row-name is used in a report with only one row-name column, only the rightmost part of the name appears in the report.
- The practical length of the row name is limited by the width of the column(s) in which it is output. Characters to the right that would overwrite information in the next column are truncated.
- To store a multiple-value array into a calculated row prior to the point where you have defined your columns (with your column dimension member selections), you can use NS to pre-allocate a larger number of columns with which to work with. If you supply fewer values than there are data columns, the operation using the array stops after the last array value and there are no changes to the remaining columns based on that operator. If the extra columns are currently missing, they stay missing; if they are non-missing, they retain their current values.
- Expressions are always computed from left to right. Parentheses may not be used for grouping.
- Expressions cannot contain member names.
- Commands that designate columns must use valid data column numbers, as determined by the *original* order of the columns.
- All operators in an expression must be preceded and followed with a single space.
- Integer and floating point constants are supported in expressions as single entries or members of an array.
- Row calculations are created with three commands: CALCULATE ROW, SETROWOP, and PRINTROW.

Example

The following samples demonstrate row calculations that you can perform. Note that "Total Sales" in the examples represent a calculated row, not a member name.

To compute "Avg Sales" by dividing by the constant 2:

```
{ CALCULATE ROW "Avg Sales" = "Total Sales" / 2 }
```

To multiply the first six data columns of the calculated row "Total Sales" by the six factors and store the result in the calculated row "Factored Sales":

```
{ CALCULATE ROW "Factored Sales" = "Total Sales" * [1.0 1.3 1.9 2.3 3.0  
3.7 ] }
```

To store five factors in the first five columns of "Factors", for use in later calculated row computations and/or PRINTROW output:

```
{ CALCULATE ROW "Factors" = [ 1.3 2.6 3.1 2.3 5 ] }
```

To store the value from the seventh column of "Total Sales", multiplied by 1000, in every column of the calculated row "Ending Sales":

```
{ CALCULATE ROW "Ending Sales" = "Total Sales" 7 * 1000 }
```

To set the value in column 7 of "Ending Sales" to the corresponding value from the row "Total Sales":

```
{ CALCULATE ROW "Ending Sales"7 = "Total Sales" }
```

"Total" refers to itself in this calculation and divides itself by 1000:

```
{ CALCULATE ROW "Total" = "Total" / 1000. }
```

To show a variety of operations used in one expression, use an expression like this:

```
{ CALCULATE ROW "xyz" = [ 11 12.3 -6 ] / 7 + "abc"2 % 4300. + 10 }
```

This expression divides the three values in the array by the constant 7 (if there are currently more than three data columns, the extra columns remain #Missing), adds the value from column 2 of "abc" to every column, and computes the resulting row's values as percentages of the constant 4300, and adds the constant 10 to all columns, storing the final result in "xyz". Note that if there are more than three data columns, the result in the extra columns is 10, since prior to the last operation, they were #Missing.

See Also

- [CLEARROWCALC](#)
- [CLEARALLROWCALC](#)
- [DUPLICATE](#)
- [OFFCOLCALCS](#)
- [OFFROWCALCS](#)
- [ONCOLCALCS](#)
- [ONROWCALCS](#)
- [OUTPUT](#)
- [PRINTROW](#)
- [REMOVECOLCALCS](#)
- [RENAME](#)
- [SAVEROW](#)
- [SETROWOP](#)
- [SUPOUTPUT](#)

CHILDREN

Selects all members in the level immediately below the specified member.

This command does not select the specified member.

Syntax

```
<CHILDREN mbrName
```

Parameter Description

mbrName Dimension or member name of the parent

Notes

- If member names contain spaces (for example, Cost of Goods Sold) or consist of numbers (for example, 100-10), they must be enclosed in double quotes.
- CHILDREN lists members in their outline order. The parent, specified by *mbrName*, is not included.
- The ICHILDREN command includes the specified member.

Example

```
<CHILDREN Year
```

Selects members Qtr1, Qtr2, Qtr3, and Qtr4, in that order (see the Notes for this command).

```
<CHILD Qtr1
```

Selects members Jan, Feb, and Mar, in that order.

See Also

- [DESCENDANTS](#)
- [ICHILDREN](#)
- [IDESCENDANTS](#)

CLEARALLROWCALC

Resets the value of all calculated rows to #MISSING.

Syntax

```
{ CLEARALLROWCALC }
```

See Also

- [CALCULATE ROW](#)
- [CLEARROWCALC](#)
- [OFFCOLCALCS](#)
- [OFFROWCALCS](#)
- [ONCOLCALCS](#)
- [ONROWCALCS](#)
- [PRINTROW](#)

- [REMOVECOLCALCS](#)
- [SETROWOP](#)
- [SUPOUTPUT](#)

CLEARROWCALC

Resets the value of the row calculation *name* to #MISSING.

Syntax

```
{ CLEARROWCALC name }
```

Parameter Description

name Name of a calculated row from a CALCULATE ROW command.

See Also

- [CALCULATE ROW](#)
- [CLEARALLROWCALC](#)
- [OFFCOLCALCS](#)
- [OFFROWCALCS](#)
- [ONCOLCALCS](#)
- [ONROWCALCS](#)
- [PRINTROW](#)
- [REMOVECOLCALCS](#)
- [RENAME](#)
- [SAVEANDOUTPUT](#)
- [SAVEROW](#)
- [SETROWOP](#)
- [SUPOUTPUT](#)

COLHEADING

Turns on automatic display of the column header, and sets it to be output prior to display of the next non-suppressed output data row.

Syntax

```
{ COLHEADING }
```

Notes

- The purpose of delaying the header output is to ensure that when no data follows a heading (due to suppression with a SUPMISSING or at the end of a report, for instance, a meaningless header is not generated.
- IMMHEADING produces a new page and column heading immediately, without waiting for the next non-suppressed output line.
- COLHEADING can be specified between the STARTHEADING and ENDHEADING commands to position the heading relative to other outputs defined in the custom heading.

- When COLHEADING is used, the column members are displayed at the time the heading is generated, rather than immediately. Thus, if this command was issued at the start of the report script, it would still generate column headings only as part of the regular heading, and not as the first item on the page.
- COLHEADING also displays column headings after they have been suppressed with either a SUPCOLHEADING, SUPHEADING, or SUPALL command.
- By default, page and column headers (together called the HEADING) are turned on. This means they are displayed prior to the first actual output row in a report, and are reset to display again whenever:
 1. A new page is generated.
 2. Any member in the page or column dimensions changes.
- A specific COLHEADING, PAGEHEADING, or IMMHEADING dictates a new heading. Once they are reset to "display", they are output just prior to the new non-suppressed output row.

Example

The command COLHEADING displays the column heading members for a second time in the following report after displaying a blank line with the SKIP command.

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual
  <COLUMN (Year)
  <ICHILDREN Year
<ROW (Product)
<ICHILDREN Audio
{ SKIP COLHEADING }
<ICHILDREN Visual
!
```

This example produces the following report:

	Chicago Sales Actual				
	Qtr1	Qtr2	Qtr3	Qtr4	Year
	=====	=====	=====	=====	=====
Stereo	2,591	2,476	2,567	3,035	10,669
Compact_Disc	3,150	3,021	3,032	3,974	13,177
Audio	5,741	5,497	5,599	7,009	23,846
	Qtr1	Qtr2	Qtr3	Qtr4	Year
	=====	=====	=====	=====	=====
Television	4,410	4,001	4,934	6,261	19,606
VCR	3,879	3,579	4,276	4,877	16,611
Camera	2,506	2,522	2,602	3,227	10,857
Visual	10,795	10,102	11,812	14,365	47,074

See Also

- [HEADING](#)
- [SUPCOLHEADING](#)
- [IMMHEADING](#)
- [SUPPAGEHEADING](#)
- [PAGEHEADING](#)

- [TEXT](#)

COLUMN

Defines the dimensions displayed as column members. Column members are displayed above data columns.

The order of the members in the command determines the order of the column headers in the report. The first header line of column members are from the same dimension as the first member in the *dimList*. The second line members are from the dimension of the second member, and so on. *DimList* can contain a maximum of one member from each dimension.

Once you have identified the column dimensions using this command, any members from those dimensions that are a part of the report are defined as the data columns. If a member is not selected from a column dimension, then the highest member in that dimension is used.

Syntax

```
<COLUMN ( dimList )
```

Parameter Description

dimList Dimension name or a comma-delimited list of dimensions

Notes

- If dimension names contain spaces or consist of numbers, they must be enclosed in double quotes.
- When more than one dimension is specified, the first dimension in the list appears at the top of each column, the next dimension in the list appears lower on the page, nested below the first dimension, and so on.
- The maximum number of members in a dimension that is being specified for reporting on Hybrid Analysis is 255.

Example

```
<COLUMN (Year, Scenario)
```

Creates a report with Year members at the head of each column. Nested below each Year member are columns headed by members of Scenario.

See Also

- [PAGE](#)
- [ROW](#)

COMMAS

Displays commas for numbers greater than 999 after commas have been suppressed with either a SUPCOMMAS or SUPALL command.

Syntax

```
{ COMMAS }
```

Example

```
{ COMMAS }
```

displays the number 1345 as 1,345 in the report.

See Also

- [BRACKETS](#)
- [DECIMAL](#)
- [SUPALL](#)
- [SUPCOMMAS](#)

CURHEADING

Enables the display of the currency conversion heading.

Syntax

```
{ CURHEADING }
```

Notes

This command turns on the display of the currency conversion heading, if it was suppressed with SUPCURHEADING. The currency conversion heading is displayed along with each page heading as it is displayed.

Example

See the example for the [CURRENCY](#) command.

See Also

- [IMMHEADING](#)
- [CURRENCY](#)
- [SUPCURHEADING](#)
- [TEXT](#)

CURRENCY

Converts data values in the report to the *targetCurrency*, and causes the currency heading to be displayed with the page heading. This does not convert the data in the database: only in the report.

If the <CURRENCY command is not used, the data is reported as it is currently stored in the database. Typically, the database is set up with currency conversions, requiring no additional conversion.

Syntax

```
<CURRENCY targetCurrency
```


Parameter	Description
-----------	-------------

targetCurrency	Currency and currency type to display in the report. Currency type is optional. Up to four members (at most one from each currency database dimension) in a cross-dimensional member (->) For example:USD, or USD->Actual->Jun99
----------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Notes

- The currency conversion label, which identifies the currency used in the report, appears at the top of each page. See the [TEXT](#) command for custom placement of the currency label.
- For information on creating and maintaining currency databases, see the *Oracle Essbase Database Administrator's Guide*.

Example

```
<PAGE (Market, Measures, Scenario)
Illinois Sales Budget
<COLUMN (Year)
<CHILDREN Qtr1
<CURRENCY USD
<ICHILDREN Colas
!
```

This example produces the following report:

Currency: USD

	Illinois Sales Budget		
	Jan	Feb	Mar
	=====	=====	=====
100-10	360	370	380
100-20	240	260	280
100-30	#Missing	#Missing	#Missing
100	600	630	660

See Also

- [CURHEADING](#)
- [SUPCURHEADING](#)
- [TEXT](#)

DATEFORMAT

Report Writer can be used prepare reports based on Date type members. Report writer display format directives that apply to numeric values apply to Date type values also. The following format directive formats all the output cells based on the outline's date format string:

```
{OUTFORMATTEDVALUES}
```

Report Writer post-processing commands that operate on numeric data, such as like RESTRICT, TOP, BOTTOM and SORT, will operate on internal numeric date values.

Syntax

```
{ DATEFORMAT "string" }
```

Parameter Description

“string” A string in one of the date format string supported by Essbase

Example

See Also

- [WITHATTR](#)

DECIMAL

Determines the number of decimal places to display in the report.

Syntax

```
{ DECIMAL decPlaces | VARIABLE [ columnN [columnN] ] }
```

Parameter Description

decPlaces Number of decimal places to display. Positive integer from 0 (the default) to 40. Specify either VARIABLE or *decPlaces*.

VARIABLE Allows the decimal to float; may switch to scientific notation (E+00 format) if necessary to display the significant digits of a number in the given column width.

columnN **Optional.** Space-separated list of columns to be affected. If omitted, all columns are affected.

Notes

If you specify columns in the DECIMAL command *before* designating them with a member selection, the column numbers apply to all selected columns with a number that is a *multiple* of the specified column number.

The total number of specified column numbers should not exceed the value of *columnN*.

Default Value

Positive integer from 0 (the default) to 40.

Example

```
{DECIMAL 2}
```

Displays the number 65.4365 as 65.44 in the final report.

See Also

- [BRACKETS](#)
- [COMMAS](#)
- [SUPBRACKETS](#)
- [SUPCOMMAS](#)

DESCENDANTS

Adds the descendants of *mbrName* to the report, excluding *mbrName*.

Adding the descendants of the top of the dimension adds all the members in the dimension to the report, except the dimension top.

When a generation or level name is provided, this command returns all descendants at (or up to) the specified generation or level below *mbrName*.

Syntax

```
<DESCENDANTS mbrName
```

When used as an extraction command in conjunction with the < LINK command, the syntax is:

```
<DESCENDANTS (mbrName [, gen/levelName [, AT|UPTO]])
```

Parameter	Description
<i>mbrName</i>	Name of parent of descendants.
<i>gen/levelName</i>	Optional. Generation or level name.
AT	Optional. Keyword indicating that all descendants at the specified generation or level should be returned. If AT or UPTO are omitted, this behavior is the default.
UPTO	Optional. Keyword indicating that all descendants between the root member and up to the specified generation or level should be returned. The root member is also returned.

Notes

- The IDESCENDANTS command includes the specified member.
- The DESCENDANTS command, when used with UPTO keyword, includes the specified member.
- Syntax specifying generation or level is available only when this command is used as an extraction command in conjunction with the <LINK command.

Example

Example 1 (DESCENDANTS)

```
<DESCENDANTS Year
```

Selects members Jan, Feb, Mar, Q1, Apr, May, June, Q2, Jul, Aug, Sep, Q3, Oct, Nov, Dec, Q4.

Example 2 (DESCENDANTS)

```
<LINK (<DESCENDANTS (Market, "Lev0,Market" )  
OR  
<LINK (<DESCENDANTS (Market, State) )  
!
```

This example produces the following report:

New York	#Missing
Massachusetts	#Missing
Florida	#Missing
Connecticut	#Missing
New Hampshire	#Missing
California	#Missing
Oregon	#Missing
Washington	#Missing
Utah	#Missing
Nevada	#Missing
Texas	#Missing
Oklahoma	#Missing
Louisiana	#Missing
New Mexico	#Missing
Illinois	#Missing
Ohio	#Missing
Wisconsin	#Missing
Missouri	#Missing
Iowa	#Missing
Colorado	#Missing

Example 3 (DESCENDANTS)

```
<LINK (<DESCENDANTS (Market, "Lev0,Market", UPTO) )
OR
<LINK (<DESCENDANTS (Market, State, UPTO) )
!
```

This example produces the following report:

Market	#Missing
New York	#Missing
Massachusetts	#Missing
Florida	#Missing
Connecticut	#Missing
New Hampshire	#Missing
East	#Missing
California	#Missing
Oregon	#Missing
Washington	#Missing
Utah	#Missing
Nevada	#Missing
West	#Missing
Texas	#Missing
Oklahoma	#Missing
Louisiana	#Missing
New Mexico	#Missing
South	#Missing
Illinois	#Missing
Ohio	#Missing
Wisconsin	#Missing
Missouri	#Missing
Iowa	#Missing
Colorado	#Missing
Central	#Missing

See Also

- [CHILDREN](#)
- [ICHILDREN](#)
- [IDESCENDANTS](#)
- [LINK](#)

DIMBOTTOM

Adds all level-0 dimension members to the report, if the dimension is not Hybrid Analysis-enabled. In a Hybrid Analysis-enabled dimension, DIMBOTTOM adds the bottom members in a relational source only.

Syntax

```
<DIMBOTTOM mbrName
```

Parameter Description

mbrName A member from the dimension.

Notes

This command adds all level 0 members to the report. *mbrName* is from the dimension whose level 0 members you want to select. Regardless of the member you specify, Essbase retrieves all level 0 members of that dimension. For example, if you specify Audio in the Demo Basic database, Essbase retrieves all the level 0 members under Audio and under Visual, because they are all level 0 members of the Product dimension.

Example

The command <DIMBOTTOM Audio adds all the members from the bottom of the Product dimension:

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual
    <COLUMN (Year)
        <ICHILDREN Year
<ROW (Product)
<DIMBOTTOM Audio
    !
```

This example produces the following report:

	Chicago Sales Actual				
	Qtr1	Qtr2	Qtr3	Qtr4	Year
	=====	=====	=====	=====	=====
Stereo	2,591	2,476	2,567	3,035	10,669
Compact_Disc	3,150	3,021	3,032	3,974	13,177
Television	4,410	4,001	4,934	6,261	19,606
VCR	3,879	3,579	4,276	4,877	16,611
Camera	2,506	2,522	2,602	3,227	10,857

See Also

- [DIMTOP](#)

DIMEND

Specifies a dimension format to be processed after cycling through all members in the dimension.

Any formatting commands in the report script encountered immediately before the DIMEND command become formats for all dimensions in *dimList*.

When the report is produced, after processing all members from the specified dimension(s) associated with the format, including the processing of any groups of members from other dimensions which are nested inside the specified dimension(s), the DIMEND format is then processed.

Syntax

```
<DIMEND dimList
```

Parameter Description

dimList List of members, separated by commas, that represents the dimensions for which the format is intended.

Notes

Formats are associated with the subsequent member, and are processed just prior to any output of that member. Therefore, without this command, in some situations it would be impossible to define a format to process after a member, especially if it was the last in a group.

Example

The UCOLUMNS format command underlines the columns in the report after every cycle through the Market dimension. In the report, you see children of Qtr1 for East followed by children of Qtr1 for West. After West, before starting over with East again, the processing of UCOLUMNS displays the underlines in the report.

```
<PAGE (Accounts, Scenario)
Sales Actual
  <COLUMN (Product)
    /* Applied after dimension processing*/
  <CHILDREN Audio
<ROW (Market,Year)
  East West
  <CHILDREN Qtr1
{ UCOLUMNS }
<DIMEND(Market)
/* Puts underline after Market */
!
```

This example produces the following report:

```
                Sales Actual
Stereo   Compact   Audio
```

East	Jan	2,788	3,550	6,338
	Feb	2,482	3,285	5,767
	Mar	2,569	3,458	6,027
West	Jan	4,102	4,886	8,988
	Feb	3,723	4,647	8,370
	Mar	3,808	4,788	8,596

DIMTOP

Adds the top of the dimension for the member to the report.

Syntax

```
<DIMTOP mbrName
```

Parameter Description

mbrName Single member from the dimension to designate.

Notes

You can specify any member from the dimension, including the top member.

Example

```
<DIMTOP Stereo
```

Adds the top of the Product dimension to the report.

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual
```

```
<COLUMN (Year)
<CHILDREN Year
```

```
<ROW (Product)
<DIMTOP Stereo
!
```

This example produces the following report:

	Chicago Sales Actual				
	Qtr1	Qtr2	Qtr3	Qtr4	Year
	=====	=====	=====	=====	=====
Product	16,536	15,599	17,411	21,374	70,920

See Also

- [DIMBOTTOM](#)
- [DIMEND](#)

DUPLICATE

Enables a member name to occur more than once in a dimension group selection.

This command is useful either (a) in a multi-section report when the same row name appears more than once in each section or (b) when the row must be captured (without printing) once at the top of each section for calculation purposes, and included again later in the section for output.

Syntax

```
<DUPLICATE mbrRange
```

Parameter Description

mbrRange A single member name or selection command.

- Single member: A member already selected for the dimension can be selected again.
- Selection command: <DUPLICATE applies to all members selected by *mbrRange*. For example, <CHILDREN Accounts.

Notes

- If the DUPLICATE command is not used, by default the data extraction operation ignores duplicates in a group of members in the same dimension up to the point where a "!" is encountered.
- <DUPLICATE is not restricted to row dimensions. It can also be used to allow a repeat of a column or page dimension member.

Example

The following example is based on Sample Demo.

```
<PAGE (Market)
East
        <COLUMN (Scenario, Year)
                Budget Actual
                Jan    Jan

{ ORDER 2,0,1,3,4 WIDTH 12 0 1 NOINDENTGEN AFTER "%" 4
  SKIPONDIM Product LMARGIN 10
}
<ROW (Product, Accounts)

{ CALC ROW "Sales" OFF }
{ CALC COL "Actual~% Sales" = 2 % "Sales" 2 }

<ICHILDREN Visual

{ SAVEROW }   Sales
  Payroll
  Marketing
  Profit

<DUPLICATE Sales
```


!

This example produces the following report:

Budget			East	
Jan			Actual	Actual
=====			Jan	% Sales
=====			=====	=====
1,200	Television	Payroll	1,236	25%
440		Marketing	365	9%
1,240		Profit	1,295	26%
4,800		Sales	5,244	100%
1,030	VCR	Payroll	1,044	25%
150		Marketing	156	4%
1,466		Profit	1,417	35%
4,200		Sales	4,311	100%
1,195	Camera	Payroll	1,167	42%
300		Marketing	288	11%
528		Profit	400	19%
2,850		Sales	2,656	100%
3,425	Visual	Payroll	3,447	29%
890		Marketing	809	8%
3,234		Profit	3,112	27%
11,850		Sales	12,211	100%

See Also

- [PAGE](#)
- [COLUMN](#)
- [ROW](#)

ENDHEADING

Ends the definition of the custom page heading displayed at the top of each page.

Syntax

```
{ ENDHEADING }
```

Notes

This command ends the definition of the custom page heading displayed at the top of each page in the report and in certain other situations. The `STARTHEADING` command begins the heading, and all commands encountered between the `STARTHEADING` and `ENDHEADING` are part of the heading definition.

Example

See example for the [STARTHEADING](#) command.

See Also

- [HEADING](#)
- [IMMHEADING](#)
- [STARTHEADING](#)
- [SUPHEADING](#)

EUROPEAN

Enables non-US number formatting by switching commas and decimal points in report data values.

Syntax

```
{ EUROPEAN }
```

Notes

In non-US number formatting, decimal points are used as the thousands separator, while commas separate the decimal from the integer.

Example

The following example is based on Sample Demo.

This report displays an example of the { EUROPEAN } command for the report based on Chicago followed by the { SUPEUROPEAN } command for the Boston report.

```
<PAGE(Market, Accounts, Scenario)
Chicago Sales Actual
  <COLUMN (Year)
  <CHILDREN Year
```

```
<ROW (Product)
<CHILDREN Audio
  !
```

```
{EUROPEAN}
```

```
Chicago Sales Actual
```

```
  <CHILDREN Year
  <CHILDREN Audio
  !
```

This example produces the following report:

```
                Chicago Sales Actual
                Qtr1      Qtr2      Qtr3      Qtr4
                =====  =====  =====  =====
Stereo          2,591     2,476     2,567     3,035
Compact_Disc   3,150     3,021     3,032     3,974
```

```
                Chicago Sales Actual
                Qtr1      Qtr2      Qtr3      Qtr4
                =====  =====  =====  =====
```

Stereo	2.591	2.476	2.567	3.035
Compact_Disc	3.150	3.021	3.032	3.974

See Also

- [BRACKETS](#)
- [COMMAS](#)
- [DECIMAL](#)
- [SUPBRACKETS](#)
- [SUPCOMMAS](#)
- [SUPEUROPEAN](#)

FEEDON

Enables page break insertion when the number of output lines on a page is greater than the PAGELENGTH setting.

Syntax

```
{ FEEDON }
```

Notes

This command enables page breaks (and, by default, a new page header) in a report when the number of output lines on a page is greater than the PAGELENGTH setting. Use after a SUPFEED command has disabled page breaks.

Default Value

The defaults are FEEDON and PAGELENGTH of 66 lines.

See Also

- [PAGELENGTH](#)
- [SUPFEED](#)

FIXCOLUMNS

Fixes the number of columns in the report regardless of how many columns are originally selected.

Syntax

```
{ FIXCOLUMNS number }
```

Parameter Description

number Number of columns that you want to be displayed in your final report.

Notes

This command fixes the number of columns in the final report regardless of how many columns are originally selected. Only the first *number* of columns, which includes row name columns and data columns, are displayed.

This command is often used in conjunction with the ORDER command to select and reorder a subset of columns, cutting off excess columns.

Example

The following examples are based on Sample Demo.

The command { FIXCOLUMNS 3 } causes only 3 columns, the row name column and two data columns, to be displayed even though there are additional columns for the data values of Qtr3, Qtr4 and Year.

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual

      <COLUMN (Year)
      <CHILDREN Year

<ROW (Product)

{FIXCOLUMNS 3}
<CHILDREN Audio
!
```

This example produces the following report:

```
      Chicago Sales Actual

      Qtr1      Qtr2
      =====
Stereo      2,591      2,476
Compact_Disc 3,150      3,021
  Audio      5,741      5,497
```

This example used FIXCOLUMNS and ORDER to create a non-symmetric report.

```
<PAGE (Market, Accounts)
<COLUMN (Year, Scenario)
<ROW (Product)
{ ORDER 0,1,3,5,6 FIXCOLUMNS 5 }

Chicago Sales

      Jan Feb Mar
      Actual Budget

<CHILDREN Audio
!
```

Chicago Sales

	Jan	Feb	Mar	Mar
	Actual	Actual	Actual	Budget
	=====	=====	=====	=====
Stereo	923	834	834	900
Compact_Disc	1,120	1,050	980	1,000
Audio	2,043	1,884	1,814	1,900

If the command { BLOCKHEADERS } had also been used, the output would be:

Chicago Sales

	Jan	Feb	Mar	Mar
	Actual	Actual	Actual	Budget
	=====	=====	=====	=====
Stereo	2,591	2,476	2,348	2,438
Compact_Disc	3,150	3,021	3,115	3,028
Audio	5,741	5,497	5,825	5,003

Note that without the FIXCOLUMNS, the column headers would have been:

	Jan	Feb	Mar
	Actual Budget	Actual Budget	Actual Budget

See Also

- [ORDER](#)

FORMATCOLUMNS

Expands the number of data columns when processed.

Syntax

```
{ FORMATCOLUMNS number }
```

Parameter Description

number Expected number of columns that are encountered for formatting purposes.

Notes

Before any data column members are added, the report assumes only one data column. FORMATCOLUMNS (and other commands that reference column numbers) expands the number of data columns. FORMATCOLUMNS formats the report layout for a predetermined *number* of data columns for text and headings.

This command does not limit the number of output columns, as FIXCOLUMNS does. For example, a TEXT command used to center text can be issued before the addition of members that define the data columns, so centering would be off unless FORMATCOLUMNS is used to indicate the expected number of columns.

Example

{ FORMATCOLUMNS 10 } sets up an expected report size of 10 columns for formatting purposes.

See Also

- [COLUMN](#)
- [NAMESCOL](#)

GEN

Returns all members in a dimension with the specified generation name. This command does not work with Hybrid Analysis members.

Syntax

GEN name, dimension

When used as an extraction command in conjunction with the <LINK command, the syntax is:

<GEN (dimension, genNumber)

Parameter	Description
-----------	-------------

name	Generation name
------	-----------------

dimension	Dimension name
-----------	----------------

genNumber	Generation number
-----------	-------------------

Notes

- The report script can use either default generation names or user-defined generation names. Examples of default generation names are GEN1, GEN2, and so on.
- Use quotes around the GEN command if the dimension name contains spaces.

Example

GEN3, Year

Selects members of generation 3 from the Year dimension.

CityGen, State

Selects members of the user-defined generation name CityGen from the State dimension.

"GEN2, All Markets"

Selects members of generation 2 from the All Markets dimension.

<LINK (<GEN(Product, 3) AND <LEV(Product, 0))

Selects members with generation 3 and level 0 from the Product dimension.

See Also

- [LEV](#)

- [LINK](#)

HYBRIDANALYSISON

Enables Essbase to interpret Report Writer commands in the context of a Hybrid Analysis relational source.

Syntax

```
<HYBRIDANALYSISON
```

Notes

- If a database contains a Hybrid Analysis relational source, specifying the <HYBRIDANALYSISON command before Report Writer commands causes Essbase to determine if the Report Writer commands need to be extended to include one or more Hybrid Analysis members.
- You can use the <HYBRIDANALYSISON command multiple times in a report, and you can place the command around members or expansions that do not include Hybrid Analysis members without affecting the report results. <HYBRIDANALYSISON applies to all subsequent member selections until its effect is cancelled by the <HYBRIDANALYSISOFF command.
- Disabling Hybrid Analysis in Administration Services takes precedence over the <HYBRIDANALYSISON setting for any dimensions that are disabled and thus cancels any <HYBRIDANALYSISON command issued in Report Writer.
- Reports that include Hybrid Analysis members are always symmetric. Specifying the <ASYM command in a report script results in an error.
- ALLINSAMEDIM is not supported with HYBRIDANALYSISON.

Example

Assume that some members of the Product dimension are present in a Hybrid Analysis relational source and that no members of Scenario and Market are present in the Hybrid Analysis relational source. When the following report script is run, Hybrid Analysis members in the Product dimension are returned.

```
<DESC "Scenario"  
<HYBRIDANALYSISON  
<DESC "Product"  
<DESC "Market"
```

Assume that 100-10-1 is present in the Hybrid Analysis relational source. The following report script produces an error because 100-10-1 does not exist in the Essbase outline. If <HYBRIDANALYSISON were specified, 100-10-1 would be recognized in the report.

```
<DESC "Scenario"  
<DESC "100-10-1"  
<DESC "Market"
```

See Also

- [HYBRIDANALYSISOFF](#)

- [ASYM](#)
- [SYM](#)

HYBRIDANALYSISOFF

Prevents Essbase from interpreting Report Writer commands in the context of a Hybrid Analysis Relational Source.

Syntax

```
<HYBRIDANALYSISOFF
```

Notes

If a database contains a Hybrid Analysis Relational Source, specifying the <HYBRIDANALYSISOFF command before one or more Report Writer commands prevents Essbase from extending the report to include Hybrid Analysis members. You can use <HYBRIDANALYSISOFF multiple times in a report. <HYBRIDANALYSISOFF applies to all subsequent member selections until its effect is cancelled by the <HYBRIDANALYSISON command.

Example

Assume that some members of the Product and Market dimensions are present in the Hybrid Analysis Relational Source and that no members of Scenario are present in the Hybrid Analysis Relational Source. When the following report script is run, Hybrid Analysis members in the Product dimension are returned; however, Hybrid Analysis members in the Market dimension are not returned because retrievals subsequent to the <HYBRIDANALYSISOFF command cannot include a dimension that has members present in the Hybrid Analysis Relational Source.

```
<DESC "Scenario"  
<HYBRIDANALYSISON  
<DESC "Product"  
<HYBRIDANALYSISOFF  
<DESC "Market"
```

See Also

- [HYBRIDANALYSISON](#)
- [ASYM](#)
- [SYM](#)

HEADING

Displays the page heading: either the default heading or the heading as defined with the STARTHEADING and ENDHEADING commands.

If the SUPHEADING command has been used to turn off the display of the heading, this command also turns it back on, printing it just before the next non-suppressed output row, and thereafter at the top of every new page (unless SUPHEADING is used again). The heading

automatically adjusts to any change in column or page selection members and is generated prior to the next output data row without the need for a further HEADING command.

Note: The default heading includes the page member heading, the column member heading, and, if applicable, the currency heading.

Syntax

```
{ HEADING }
```

Notes

- By default, page and column headers (together called the HEADING) are turned on. This means they are displayed prior to the first actual output row in a report, and are reset to display again whenever:
 - A new page is generated.
 - Any member in the page or column dimensions changes.
 - A specific COLHEADING, PAGEHEADING, or IMMHEADING dictates a new heading. Once they are reset to "display", they are output just prior to the new non-suppressed output row.
- To produce a new page and column heading immediately, without waiting for the next non-suppressed output line, use IMMHEADING.
- A heading normally comprises the page heading (members of the PAGE dimension) and the column heading (the current members of the column dimensions). The last line of the column header is also underlined.
- If STARTHEADING/ENDHEADING is used, the HEADING command redefines the makeup of the report heading.
- If SUPHEADING is used, the page heading and column heading can still be independently turned back on by the commands: PAGEHEADING and COLHEADING.

Example

See the example for the [STARTHEADING](#) command for an example of a heading.

See Also

- [COLHEADING](#)
- [ENDHEADING](#)
- [IMMHEADING](#)
- [PAGEHEADING](#)
- [STARTHEADING](#)
- [SUPHEADING](#)

IANCESTORS

Adds a member and its ancestors to the report.

Syntax

<IANCESTORS *mbrName*

Parameter Description

mbrName Single member whose ancestors you want to include.

Notes

The ancestors of a member consists of its parent, that parent's parent, and so on, all the way to the top member of the dimension, including the specified member.

See Also

- [CHILDREN](#)
- [DESCENDANTS](#)
- [PARENT](#)

ICHILDREN

Selects the specified member and all members in the level immediately below it.

Syntax

<ICHILDREN *mbrName*

Parameter Description

mbrName Dimension or member name of the parent

Notes

- If member names contain spaces (for example, Cost of Goods Sold or consist of numbers (for example, 100-10), they must be enclosed in double quotes.
- ICHILDREN lists members in their defined order, according to the database outline. The parent, which is the member specified as the parameter in the ICHILDREN command, is listed last.

Example

<ICHILDREN Year

Selects members Qtr1, Qtr2, Qtr3, Qtr4, and Year, in that order.

<ICHILDREN Qtr1

Selects members Jan, Feb, Mar, and Qtr1, in that order.

See Also

- [ANCESTORS](#)
- [CHILDREN](#)
- [DESCENDANTS](#)
- [PARENT](#)

IDESCENDANTS

Adds the specified member and its descendants to the report.

Syntax

```
<IDESCENDANTS mbrName
```

When used as an extraction command in conjunction with the <LINK command, the syntax is:

```
<IDESCENDANTS (mbrName [, gen/levelName [, AT|UPTO]])
```

Parameter	Description
mbrName	Name of single member and descendants to add to the report.
gen/levelName	Optional. Generation or level name.
AT	Optional. Keyword indicating that all descendants at the specified generation or level should be returned. If AT or UPTO are omitted, this behavior is the default.
UPTO	Optional. Keyword indicating that all descendants between the root member and up to the specified generation or level should be returned. The root member is also returned.

Notes

Adding the descendants of the top of the dimension adds all the members in the dimension to the report, including the dimension top.

Example

Example 1

```
<IDESCENDANTS Product
```

Adds all the members from the Product dimension to the report since all the members are descendants of the member Product which is the top of the dimension. Audio and Visual are the children of Product. Stereo and Compact_Disc are the children of Audio while Television, VCR, and Camera are the children of Visual.

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual
<COLUMN (Year)
<CHILDREN Year
<ROW (Product)
<IDESCENDANTS Product
!
```

This example produces the following report:

	Chicago Sales Actual				
	Qtr1	Qtr2	Qtr3	Qtr4	Year
	=====	=====	=====	=====	=====
Stereo	2,591	2,476	2,567	3,035	10,669
Compact_Disc	3,150	3,021	3,032	3,974	13,177

Audio	5,741	5,497	5,599	7,009	23,846
Television	4,410	4,001	4,934	6,261	19,606
VCR	3,879	3,579	4,276	4,877	16,611
Camera	2,506	2,522	2,602	3,227	10,857
Visual	10,795	10,102	11,812	14,365	47,074
Product	16,536	15,599	17,411	21,374	70,920

Example 2

```
<LINK (<IDESCENDANTS (Market, "Lev0,Market" ) )
OR
<LINK (<IDESCENDANTS (Market,State) )
!
```

This example produces the following report:

New York	#Missing
Massachusetts	#Missing
Florida	#Missing
Connecticut	#Missing
New Hampshire	#Missing
California	#Missing
Oregon	#Missing
Washington	#Missing
Utah	#Missing
Nevada	#Missing
Texas	#Missing
Oklahoma	#Missing
Louisiana	#Missing
New Mexico	#Missing
Illinois	#Missing
Ohio	#Missing
Wisconsin	#Missing
Missouri	#Missing
Iowa	#Missing
Colorado	#Missing
Market	#Missing

Example 3

```
<LINK (<IDESCENDANTS (Market, "Lev0,Market" ,UPTO) )
OR
<LINK (<IDESCENDANTS (Market,State,UPTO) )
!
```

This example produces the following report:

Market	#Missing
New York	#Missing
Massachusetts	#Missing
Florida	#Missing
Connecticut	#Missing
New Hampshire	#Missing
East	#Missing
California	#Missing
Oregon	#Missing
Washington	#Missing

Utah	#Missing
Nevada	#Missing
West	#Missing
Texas	#Missing
Oklahoma	#Missing
Louisiana	#Missing
New Mexico	#Missing
South	#Missing
Illinois	#Missing
Ohio	#Missing
Wisconsin	#Missing
Missouri	#Missing
Iowa	#Missing
Colorado	#Missing
Central	#Missing

See Also

- [ANCESTORS](#)
- [CHILDREN](#)
- [DESCENDANTS](#)
- [PARENT](#)
- [LINK](#)

IMMHEADING

Forces the immediate display of the heading without waiting for the next non-suppressed data row.

Syntax

```
{ IMMHEADING }
```

Notes

Under normal circumstances, the heading only appears when at least one non-suppressed row is ready to be output on the current page. For this reason, when any suppression commands are turned on (such as SUPMISSING or SUPZEROS), and an entire page is suppressed, those page headers are normally skipped entirely.

An occurrence of the IMMHEADING command prints the header immediately, even if there is no current row to print. This command does not unsuppress data, but simply prints its headings.

This command is useful for inserting special formatting between the heading and the first output record. This is usually impossible because the header does not print until it is ready to output data immediately, that is, after any formats associated with the row have been processed.

Example

See the example for [STARTHEADING](#) for an example of a heading.

See Also

- [ENDHEADING](#)
- [HEADING](#)

- [STARTHEADING](#)
- [SUPHEADING](#)

INCEMPTYROWS

Displays empty rows of data, or rows that contain only zeros or #MISSING data values, in the final report.

Syntax

```
{ INCEMPTYROWS }
```

Notes

This command displays empty rows of data, or rows that contain only zeros or #MISSING data values, in the final report. This command is used to cancel the effects of SUPEMPTYROWS, SUPMISSINGROWS or SUPZEROROWS.

See Also

- [INCMISSINGROWS](#)
- [INCZEROROWS](#)
- [SUPALL](#)
- [SUPEMPTYROWS](#)
- [SUPMISSINGROWS](#)
- [SUPZEROROWS](#)

INCFORMATS

Controls the formats affected by the following commands: SUPMASK, SUPMISSING, and SUPZERO.

Syntax

```
{ INCFORMATS }
```

Notes

INCFORMATS prints out the format associated with a particular data row even when that row is suppressed. This means that line formatting, TEXT and MASK commands, and headers do not print unless their associated data rows print (or are not suppressed).

Default Value

Whenever the SUPMASK, SUPMISSING, or SUPZERO commands are used, by default SUPFORMATS is also set on, unless it has been specifically turned off.

See Also

- [SUPFORMATS](#)

INCMASK

Re-includes (turns back on) the mask that has been suppressed by the command SUPMASK.

Syntax

```
{ INCMASK }
```

See Also

- [MASK](#)

INCMISSINGROWS

Displays missing rows of data, or rows that contain all #MISSING data values, in the final report.

Syntax

```
{ INCMISSINGROWS }
```

Notes

This command displays missing rows of data, or rows that contain all #MISSING data values, in the final report. This command is used after a SUPMISSINGROWS or SUPEMPTYROWS command has been used to remove the missing rows from the final report.

See Also

- [INCEMPTYROWS](#)
- [INCZEROROWS](#)
- [SUPALL](#)
- [SUPEMPTYROWS](#)
- [SUPMISSINGROWS](#)
- [SUPZEROROWS](#)

INCZEROROWS

Includes rows that contain only data values of zero in the final report.

Syntax

```
{ INCZEROROWS }
```

Notes

This command displays zero rows of data, or rows that contain only data values of zero, in the final report. This command is used after a SUPZEROROWS or SUPEMPTYROWS command has been used to remove the zero rows from the final report.

See Also

- [INCEMPTYROWS](#)
- [INCMISSINGROWS](#)
- [SUPALL](#)

- [SUPEMPTYROWS](#)
- [SUPMISSINGROWS](#)
- [SUPZEROROWS](#)

INDENT

Shifts the first row names column in column-output order by the specified number of characters.

Note: Default (No value): Indents columns by 2.

Syntax

```
{ INDENT [ offset ] }
```

Parameter Description

- | | |
|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| offset | <p>Optional. Number of spaces to indent column 0 from the left boundary of the name column. Values:</p> <ul style="list-style-type: none"> ● Positive number (up to 100): Shifts column 0 to the right. ● Negative number: Shifts column left, but cannot indent to the left of the start of the name column. ● 0: Returns column to original position. ● Default (no value): Indents columns by 2. |
|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Notes

- { INDENT } shifts column 0 two characters to the right (the default) and decreases the size of column 1 by two.
- { INDENT 0 } resets the indent position to the original position regardless of the current position.
- When a member is indented, the width of the names column for that member is decreased to offset the indent. This does not shift the remaining columns in the report.
- Once the indented names column has been declared, you can use the ORDER command to moved it within the final output format or precede it with regular or calculated columns.
- Hierarchical relationships between row members are, by default, indicated by indentation. Indentation only applies to a group of rows generated together, such as when a single ! is used. If each consecutive row is generated independently, using its own !, then no indentation occurs.

Example

In the following example, the first report for Chicago shows the default indentation while the second report for Boston uses the { INDENT 10} command to shift the row names column 10 places to the right.

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual
```

```
<COLUMN (Year)
<ICHILDREN Year
```



```

<ROW (Product)
<CHILDREN Audio
    !

{ INDENT 10 }
Boston Sales Actual

<CHILDREN Year
<CHILDREN Audio
    !

```

This example produces the following report:

	Chicago Sales Actual				
	Qtr1	Qtr2	Qtr3	Qtr4	Year
	=====	=====	=====	=====	=====
Stereo	2,591	2,476	2,567	3,035	10,669
Compact_Disc	3,150	3,021	3,032	3,974	13,177
Audio	5,741	5,497	5,599	7,009	23,846

	Boston Sales Actual				
	Qtr1	Qtr2	Qtr3	Qtr4	Year
	=====	=====	=====	=====	=====
Stereo	2,450	2,341	2,377	2,917	10,085
Compact_~	3,290	3,034	3,132	3,571	13,027
Audio	5,740	5,375	5,509	6,488	23,112

See Also

- [INDENTGEN](#)
- [LMARGIN](#)
- [NOINDENTGEN](#)

INDENTGEN

Indents subsequent row members in the row names column based on the generation in the database outline.

Syntax

```
{ INDENTGEN [ offset ] }
```

Parameter Description

offset **Optional.** Number that determines the amount to indent each succeeding generation from the previous generation. Default: INDENTGEN -2.

Notes

This command indents row members in the row names column based on the generation in the Database Outline. Generations are counted starting at the top of the dimension.

The top of the dimension is the first generation of the dimension. The children of the top are the second generation and so on. The *offset* determines how many characters each successive generation is indented. A positive number places the first generation at the leftmost position and indents each successive generation to the right. A negative number places the last generation on the left.

By default, all generations in a row group are indented by -2 for each relative generation difference. A row group is the group of row members selected before a an exclamation point (!) is encountered. If every row is generated separately (a ! after every row member) all the "groups" are one row only, and thus are not indented because there is no relative generation difference.

The indentation is based on relative rather than absolute generation differences so that if a report is working with only the lower levels of a many-level tree, all the row names do not start heavily indented, wasting column space. If *offset* is not given, it does not have a default value of -2.

Default Value

-2 is the default at the start of each report. {INDENTGEN}

Example

The following example shows the default generation indentation for the Chicago report followed by the {INDENTGEN 3} command in the Boston report.

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual

    <COLUMN (Year)
    <ICHILDREN Year

<ROW (Product)
<IDESCENDANTS Product
!

{ INDENTGEN 3 }
Boston Sales Actual

    <ICHILDREN Year

<IDESCENDANTS Product
!
```

This example produces the following report:

	Chicago Sales Actual				
	Qtr1	Qtr2	Qtr3	Qtr4	Year
	=====	=====	=====	=====	=====
Stereo	2,591	2,476	2,567	3,035	10,669
Compact_Disc	3,150	3,021	3,032	3,974	13,177
Audio	5,741	5,497	5,599	7,009	23,846
Television	4,410	4,001	4,934	6,261	19,606
VCR	3,879	3,579	4,276	4,877	16,611
Camera	2,506	2,522	2,602	3,227	10,857
Visual	10,795	10,102	11,812	14,365	47,074

Product 16,536 15,599 17,411 21,374 70,920

	Boston Sales Actual				
	Qtr1	Qtr2	Qtr3	Qtr4	Year
	=====	=====	=====	=====	=====
Stereo	2,450	2,341	2,377	2,917	10,085
Compact_Disc	3,290	3,034	3,132	3,571	13,027
Audio	5,740	5,375	5,509	6,488	23,112
Television	4,197	3,757	4,740	5,000	17,694
VCR	3,645	3,663	4,201	4,509	16,018
Camera	2,230	2,255	2,266	3,162	9,913
Visual	10,072	9,675	11,207	12,671	43,625
Product	15,812	15,050	16,716	19,159	66,737

See Also

- [INDENT](#)
- [NOINDENTGEN](#)

IPARENT

Adds the specified member and its parent to the report.

Syntax

```
IPARENT mbrName
```

Parameter Description

mbrName A single member, which must not be the top member of the dimension.

Notes

This command selects the current member and its parent, as defined in the database outline.

Example

```
<IPARENT Jan
```

Selects the member Jan and its parent member, Qtr1, in that order.

See Also

- [PARENT](#)

LATEST

Specifies a Dynamic Time Series member in a report script, which has reserved generation names that are defined in the database outline alias table (You must create a Dynamic Time Series member in the database outline before you use it in a report script.)

If you use the < LATEST syntax, the command is applied globally in the report script. If you use the *reservedName* (*mbrName*) syntax, Essbase applies the command only to the member listed in the syntax argument.

Syntax

1:

```
<LATEST mbrName
```

2:

```
<LATEST reservedName (mbrName)
```

Parameter	Description
-----------	-------------

<i>reservedName</i>	One of the following pre-defined generation names:
---------------------	----------------------------------------------------

- History-To-Date (H-T-D)
- Year-To-Date (Y-T-D)
- Season-To-Date (S-T-D)
- Period-To-Date (P-T-D)
- Quarter-To-Date (Q-T-D)
- Month-To-Date (M-T-D)
- Week-To-Date (W-T-D)
- Day-To-Date (D-T-D)

<i>mbrName</i>	The name of the level 0 member in the Time dimension.
----------------	-------------------------------------------------------

Notes

- You can create an alias table in the database and replace the predefined generation names with alias names.
- The "latest" period must be a level 0 member in the time dimension.
- Sparse retrieval optimization eliminates requested sparse members that do not have any data blocks in the database.
- You cannot use attributes as arguments.

Example

```
<LATEST May
```

or

```
Q-T-D (May)
```

LEAVES

Adds level 0 contributing descendants (descendants with non #MISSING data) for the specified member to the report. This command is equivalent to getting DESCENDANTS of *mbrName* at level-0 (for primary hierarchy) with SUPMISSINGROWS enabled for the dimension.

The Leaves command compactly describes large dimensions correlated with another dimension (many-to-many relationship) while avoiding internal expansion of members before retrieval.

Because large sets tend to be very sparse, only a few members contribute to the input member (have non #Missing values) and are returned. As a result, LEAVES consumes less memory resources than the equivalent nonempty Descendants function call, allowing for better scalability, especially in concurrent user environments.

Syntax

```
<LEAVES mbrName
```

Parameter Description

mbrName Single member whose level 0 contributing descendants should be added to the report

Notes

- This command only applies to aggregate storage databases.
- This command can only be used on rows or pages; if used on columns, an error is returned.
- This command is not supported in combination with name and alias sorting commands. Members will be returned in outline order.
- This command is not supported in combination with other selection commands for the same dimension.
- This command is not supported in combination with row and column calculation commands.

Example

```
<LEAVES ("Personal Electronics")  
!
```

This example produces the following report:

Digital Cameras	1,344,844
Camcorders	2,747,641
Photo Printers	1,325,536
Memory	2,607,186
Other Accessori~	6,475,762
Boomboxes	1,720,446
Radios	1,657,511

"Handhelds" was omitted from the result set because it has a value of #MISSING, so it does not contribute to "Personal Electronics".

See Also

- [DESCENDANTS](#)

LEV

Returns all members in a dimension with the specified level name. This command does not work with Hybrid Analysis members.

Syntax

```
LEV name, dimension
```

When used as an extraction command in conjunction with the <LINK command, the syntax is:

```
<LEV(dimension, levNumber)
```

Parameter Description

name Level name
dimension Dimension name
levNumber Level number

Notes

- The report script can use either default level names or user-defined level names. Examples of default level names are LEV0, LEV1, and so on.
- Use quotes around the LEV command if the dimension name contains spaces.

Example

LEV0, Product

Selects members of level 0 from the Product dimension.

ZipCodeLev, State

Selects members of the user-defined generation name ZipCodeLev from the State dimension.

"LEV1, All Regions"

Selects members of level 1 from the All Regions dimension.

<LINK (<GEN(Market,2) AND NOT <LEV(Market,0))

Selects members of generation 2, but not level 0 from the Market dimension.

See Also

- [GEN](#)
- [LINK](#)

LINK

Uses the AND, OR, and NOT Boolean operators, combined with extraction commands, to refine member selections. The LINK command has been extended to span into dimension levels that are located in the Hybrid Analysis portion of an Essbase cube.

Syntax

<LINK (*extractionCommand* [*operator* *extractionCommand*])

Parameter	Description
extractionCommand	<p>Any of the following extraction commands or another AND/OR expression:</p> <ul style="list-style-type: none"> <ALLINSAMEDIM (<i>member</i>) <ALLSIBLINGS (<i>member</i>) <ANCESTORS (<i>member</i>) <CHILDREN (<i>member</i>) <DESCENDANTS (<i>member</i> [, <i>gen/levelName</i> [, AT UPTO]]) <DIMBOTTOM (<i>member</i>) <DIMTOP (<i>member</i>) <IANCESTORS (<i>member</i>) <ICHILDREN (<i>member</i>) <IDESCENDANTS (<i>member</i> [, <i>gen/levelName</i> [, AT UPTO]]) <IPARENT (<i>member</i>) <MATCH (<i>Dimension</i>, <i>match_string</i>) <MEMBER (<i>member</i>) <OFSAMEGEN (<i>member</i>) <ONSAMELEVELAS (<i>member</i>) <PARENT (<i>member</i>) <UDA (<i>Dimension</i>, <i>UDA_name</i>)
Operator	<p>Any of the following Boolean operators:</p> <ul style="list-style-type: none"> ● Use the AND operator when all conditions must be met. ● Use the OR operator when either one condition or another must be met. ● Use the NOT operator to choose the inverse of the selected condition.

Notes

- NOT can only be associated with an extraction command, and does not apply to the entire expression. You must use NOT in conjunction with either the AND or OR operators.
- The MEMBER extraction command is only used within a LINK expression; you can use the MEMBER selection to select a single member. Do not use the MEMBER command outside of a LINK expression.
- You must select members from the same dimension, and all extraction command arguments must be enclosed in parentheses, as in the example above.
- Essbase evaluates operators from left to right. Use parentheses to group the expressions. For example: A OR B AND C is the same as ((A OR B) AND C). In the first expression Essbase evaluates the expression from left to right, evaluating A OR B before evaluating AND C. In the second expression, Essbase evaluates the sub-expression in parentheses (A OR B) before the whole expression, producing the same result. However, if you use (A OR (B AND C)), Essbase evaluates the sub-expression in parentheses (B AND C) before the whole expression, producing a different result.
- You can include up to 50 arguments in a LINK statement. For example, <LINK (A OR B OR (C AND D)) counts as four separate arguments.
- All extraction commands within a LINK statement need to select from the same dimensions; a command such as LINK (<ICHILDREN (east) AND <LEV (product,0)) causes a syntax error.
- The LINK command also retrieves members located in the Hybrid Analysis portion of an Essbase database. When used in conjunction with <DIMBOTTOM for a Hybrid Analysis-

enabled dimension for reports accessing data at the relational level, <DIMBOTTOM adds the bottom members in a relational source only.

- If the LINK command returns an empty set of members, nothing is returned.
- On Teradata ODBC driver version 3.0.2, using the LINK command to access Hybrid Analysis members does not work if the command uses more than one AND or OR operator. Teradata ODBC driver versions 2.7 or 3.0.3 have no issue.

Example

```
<LINK (<UDA(product,Sweet) AND <LEV(product,0))
```

Selects all level 0 products that are sweet.

```
<LINK ((<IDESCENDANTS("100") AND <UDA(product,Sweet)) OR <LEV(product, 0))
```

Selects sweet products from the "100" sub-tree plus all level 0 products.

```
<LINK ((<IDESCENDANTS("100") AND NOT <UDA(product, Sweet)) OR <LEV(product, 0))
```

Selects non sweet products from the "100" sub-tree plus all level 0 products.

See Also

- [ALLINSAMEDIM](#)
- [ALLSIBLINGS](#)
- [ANCESTORS](#)
- [CHILDREN](#)
- [DESCENDANTS](#)
- [DIMBOTTOM](#)
- [DIMTOP](#)
- [IANCESTORS](#)
- [ICHILDREN](#)
- [IDESCENDANTS](#)
- [IPARENT](#)
- [MATCH](#)
- [OFSAMEGEN](#)
- [ONSAMELEVELAS](#)
- [PARENT](#)
- [UDA](#)

LMARGIN

Sets the left margin for the report to *marginSize* characters.

Syntax

```
{ LMARGIN [ marginSize ] }
```

Parameter Description

marginSize Optional numeric value: number of character spaces for left margin.

Notes

This command sets the left margin for the report to *marginSize* characters. In most cases the value of *marginSize* should be 2 or greater when printing on a laser printer.

Default Value

If the LMARGIN command is not used, the default is 0. If *marginSize* is omitted, it assumes a default value of 0.

Example

{LMARGIN 10} sets the left margin to 10 characters.

See Also

- [INDENT](#)
- [PAGELENGTH](#)

MASK

Overwrites the text in each output row with the specified characters at the specified position.

All nonblank characters in the *text* overwrite appear in the output line.

To create a mask of a blank character that overwrites output, enter ~ (the tilde character), rather than a blank space. The ~ is output as a blank space mask.

In addition to constant text, this command can use keywords to insert special strings into the report. These keywords begin with a "*" and must be entered. These are identical to the * keywords under the TEXT command, and are listed here for convenience. For a more complete discussion of * keywords, see the [TEXT](#) command.

You may include multiple sets of *positions* and *text* in a single MASK command.

Keyword	Description
*APPNAME	Name of the application as set in the application definition.
*ARBOR	Version information from the Essbase Server.
*COLHDR <i>number1 number2</i>	Column heading members from the report, usually used with SUPCOLHEADING.
*COLHDRFULL	Full column heading, along with underlines of the column headings and a 1-line skip.
*CURRENCY	Currency conversion label that indicates to which currency the data values have been converted at report time with the CURRENCY command.
*DATE	Date the report was generated.
*DATETIME	Date and time the report was generated.
*DBNAME	Name of the data base within the application.
*EDATE	Date in European (dd/mm/yy) format.

Keyword	Description
*EDATETIME	European format date (dd/mm/yy) and time.
*MACHINE	Network name for the computer that is running the Essbase Server.
*PAGEHDR <i>number</i>	Page member heading for the report, usually used with SUPPAGEHEADING.
*PAGENO	Page number for the current page.
*PAGESTRING	Page number preceded by the text "Page:"
*TIME	Time the report was generated.
*TIMEDATE	Time and date the report was generated.
*TIMEEDATE	Time and European format (dd/mm/yy) date.
*USERNAME	Name of the user generating the report.

Syntax

```
{ MASK charPosition "replacement" [ charPosition "replacement" ] }
```

Parameter Description

charPosition Character position at which to start replacing text.

"*replacement*" New text, enclosed by quotation marks, with which to overwrite the original output.

Notes

- MASK is a setting command.
- To replace a space, use a ~ (the tilde character).
- If you want to produce an output file in comma-delimited format, use the SUPCOMMAS command, as in the example, to suppress the commas in numeric values. You can also use the SUPPAGEHEADING command to suppress page headings in the comma-delimited file.

Example

The following example is based on Sample Basic.

```
<ROW (Year, Measures, Product, Market, Scenario)
{SUPPAGEHEADING}
{ROWREPEAT}
{DECIMAL 2}
{SUPCOMMAS}
{MASK 3 " ", " 22 " , " 40 " , " 55 " , " 74 " , " }
<CHILDREN Qtr1
Sales
<CHILDREN Colas
East
Budget
!
```

This example produces the following report:

Jan,	Sales,	100-10,	East,	Budget,	1690.00
Jan,	Sales,	100-20,	East,	Budget,	190.00
Jan,	Sales,	100-30,	East,	Budget,	80.00
Feb,	Sales,	100-10,	East,	Budget,	1640.00
Feb,	Sales,	100-20,	East,	Budget,	190.00
Feb,	Sales,	100-30,	East,	Budget,	90.00
Mar,	Sales,	100-10,	East,	Budget,	1690.00
Mar,	Sales,	100-20,	East,	Budget,	200.00
Mar,	Sales,	100-30,	East,	Budget,	100.00

See Also

- [INCMASK](#)
- [SUPMASK](#)
- [TEXT](#)

MATCH

Performs wildcard member selection. Essbase searches for member names that match the pattern you specify, and returns the member names it finds.

If you defined the members names in the database you are searching as case-sensitive, the search is case-sensitive. Otherwise, the search is not case-sensitive. To define database member names as case-sensitive, use Outline Editor in Administration Services (see the *Oracle Essbase Administration Services Online Help*).

You can use more than one MATCH command in your report.

If Essbase does not find any members that match the chosen character pattern, it returns no member names and continues with the other report commands in your report.

Syntax

```
<MATCH ("Member" | "Gen" | "Level", "Pattern")
```

Parameter Description

- | | |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| "Member" | Member name at the top of the member hierarchy you want to search. Essbase searches the member name and its descendants. |
| "Gen" | Default or user-defined name of the generation you want to search. |
| "Level" | Default or user-defined name of the level you want to search. |
| "Pattern" | The character pattern you want to search for, including a wildcard character (* or ?). <ul style="list-style-type: none"> ● ? Substitutes one occurrence of any character; can be placed anywhere in the string. ● * Substitutes any number of characters; must be used at the end of the string. ● You can include spaces in the character pattern. Ensure that you enclose the pattern in quotation marks (""). |

Notes

MATCH does not function with Hybrid-Analysis relational sources.

Example

The following report is based on the Sample Basic database, and uses a * wildcard pattern search.

```
<PAGE (Measures, Market, Scenario)
Sales East Actual
<COLUMN (Year)
<MATCH (Year, J*)
<ROW (Product)
lev1,Product
!
```

Essbase searches the Year dimension and finds 3 months beginning with the letter "J":Jan, Jun, and Jul. The report returns the following data:

	Sales East Actual		
	Jan	Jun	Jul
	=====	=====	=====
100	2,105	2,625	2,735
200	1,853	2,071	1,992
300	1,609	1,795	1,926
400	1,213	1,404	1,395
Diet	620	712	778

The following report is based on the Sample Basic database, and uses a ? wildcard pattern search.

```
<PAGE (Measures, Market, Scenario)
Sales East Actual
<COLUMN (Year)
<ROW (Product)
<MATCH (Product, "???-10")
!
```

Essbase searches the Product dimension and finds all instances of products ending in "-10", and preceded by three characters. The report returns the following data:

	Sales East Actual Year
100-10	23,205
200-10	8,145
300-10	13,302
400-10	6,898

MATCHEX

Performs wildcard member selection. Essbase searches for member names that match the pattern you specify, and returns the member names it finds.

Provides an optional parameter to specify if the search should be performed on member names or aliases, regardless of whether the query output in the report script uses members or aliases.

If you defined the members names in the database you are searching as case-sensitive, the search is case-sensitive. Otherwise, the search is not case-sensitive. To define database member names

as case-sensitive, use Outline Editor in Administration Services (see the *Oracle Essbase Administration Services Online Help*).

You can use more than one MATCHEX command in your report.

If Essbase does not find any members that match the chosen character pattern, it returns no member names and continues with the other report commands in your report.

Syntax

```
<MATCH ( "Member" | "Gen" | "Level" , "Pattern" , ALT | MBR | BOTH)
```

Parameter	Description
"Member"	Member name at the top of the member hierarchy you want to search. Essbase searches the member name and its descendants.
"Gen"	Default or user-defined name of the generation you want to search.
"Level"	Default or user-defined name of the level you want to search.
"Pattern"	The character pattern you want to search for, including a wildcard character (* or ?). <ul style="list-style-type: none">● ? Substitutes one occurrence of any character; can be placed anywhere in the string.● * Substitutes any number of characters; must be used at the end of the string.● You can include spaces in the character pattern. Ensure that you enclose the pattern in quotation marks ("").
ALT MBR BOTH	Optional—The ALT MBR BOTH option overrides default pattern matching specifications. The default pattern matching uses aliases for pattern matching if aliases are to be displayed in report output, but uses names otherwise. <ul style="list-style-type: none">● ALT Filter using aliases of selected members from selected alias table for pattern matching. The alias table is set by <code>outaltselect</code>, otherwise default alias table.● MBR Filters using member names of selected members for pattern matching.● BOTH Filters using member names as well as aliases for selected members from selected alias table for pattern matching. The alias table is set by <code>outaltselect</code>, otherwise default alias table.

Notes

MATCHEX does not function with Hybrid-Analysis relational sources.

Example

```
<NewAlt " "  
<matchex(product,100,MBR)  
!  
  
<outaltselect default  
<nmatchex(product,Caff*,ALT)  
!  
  
<OUTALTSELECT "Default"  
<NewAlt "Product"
```

```
<OUTMBRNAME  
<LINK ( (<MATCHEX("Product", "100", MBR) AND < IDESCENDANTS("Product")))  
!
```

MEANINGLESTEXT

Displays #ME in place of a specified text string. Used with OUTMEANINGLESS.

Syntax

```
{ MEANINGLESTEXT "string" }
```

Parameter Description

“string” The specified string to be replaced with #ME in cells.

Example

See Also

- [WITHATTR](#)

MISSINGTEXT

Replaces the #MISSING with *text* when a missing data value is generated on a line in the report. If you do not specify *text*, the default #MISSING is restored.

Syntax

```
{MISSINGTEXT [ "text" ] }
```

Parameter Description

text Optional text to use for missing values.

Notes

- MISSINGTEXT is a setting command.
- The label must be enclosed in double quotes.

Example

```
{MISSINGTEXT "Not Applicable."}
```

See Also

- [SUPEMPTYROWS](#)
- [SUPMISSINGROWS](#)
- [SUPZEROROWS](#)
- [TEXT](#)

NAMECOL

Determines the location of the row names columns in the report.

Use the NAMECOL command *after* entering the column members in the report. You can get the same result with the ORDER command, but NAMECOL is more convenient for moving just the names columns and when the number of data columns can vary.

Syntax

```
{ NAMECOL [ columnList | CENTERED ] }
```

Parameter	Description
columnList	Optional list, separated by spaces, of the locations for each row name. List position corresponds to the number of the affected column. NAMECOL shifts the remaining columns left or right to make room for the columns of row member names.
CENTERED (or C)	Key word that centers the column of row member names in the report. Before using this parameter: <ul style="list-style-type: none">● Define all columns in the report.● Use the FORMATCOLUMNS command to set the number of columns.

Notes

{ NAMECOL c c 10 } places the first two row name columns in the center of the report, and the third row name column in column 10.

Example

The command { NAMECOL c } places the row names column in the following report in the center of the report.

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual

      <COLUMN (Year)
      <CHILDREN Year

<ROW (Product)
{ NAMECOL c }
<CHILDREN Audio
!
```

This example produces the following report:

```
Chicago Sales Actual

      Qtr1   Qtr2           Qtr3   Qtr4   Year
===== =====           ===== ===== =====
      2,591  2,476 Stereo      2,567  3,035  10,669
      3,150  3,021 Compact_Disc  3,032  3,974  13,177
      5,741  5,497 Audio        5,599  7,009  23,846
```

See Also

- [FIXCOLUMNS](#)
- [FORMATCOLUMNS](#)
- [NAMEWIDTH](#)
- [ORDER](#)

NAMESON

Turns on the display of column(s) of row member names.

Syntax

```
{ NAMESON }
```

Notes

This command reverses the effect of a SUPALL or SUPNAMES command. These commands turn off the display of column(s) of row member names in the final report.

See Also

- [SUPALL](#)
- [SUPNAMES](#)

NAMEWIDTH

Determines the width of all row name columns in the report.

Syntax

```
{ NAMEWIDTH [ width ] }
```

Parameter Description

width Optional. Specifies the total number of characters displayed for each column.

Notes

This command determines the width of the column for all row member names in the report. Member names are truncated when necessary to fit in the column and the tilde character(~) signifies that there are letters not visible in the report. If each names column needs a different width, use the WIDTH command.

Default Value

If *width* is not given, then a default value of 17 is assumed.

Example

In the following example, the first report for Chicago displays the default width for the row names column while the { NAMEWIDTH 25 } command in the Boston report increases the width of the row names column to 25.


```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual
```

```
    <COLUMN (Year)
    <CHILDREN Year
```

```
<ROW (Product)
<CHILDREN Audio
    !
```

```
{ NAMEWIDTH 25 }
```

```
Boston Sales Actual
```

```
    <CHILDREN Year
```

```
<CHILDREN Audio
    !
```

This example produces the following report:

```
                Chicago Sales Actual
                Qtr1  Qtr2  Qtr3  Qtr4  Year
                =====
Stereo          2,591  2,476  2,567  3,035  10,669
Compact_Disc   3,150  3,021  3,032  3,974  13,177

                Boston Sales Actual
                Qtr1  Qtr2  Qtr3  Qtr4  Year
                =====
Stereo          2,450  2,341  2,377  2,917  10,085
Compact_Disc   3,290  3,034  3,132  3,571  13,027
```

See Also

- [NAMECOL](#)
- [WIDTH](#)

NEWPAGE

Inserts a new page in the report regardless of how many lines have been generated for the current page.

Syntax

```
{ NEWPAGE }
```

Notes

This command inserts a new page in the report regardless of how many lines have been generated for the current page. The report continues with a new page for the next row. A new heading is displayed at the top of the new page, assuming the page has at least one non-suppressed output data row, unless SUPHEADING is used.

See Also

- [FEEDON](#)
- [SUPFEED](#)

NOINDENTGEN

Displays all row member names left-aligned in the row names column without indenting members based on generation in the database outline.

Syntax

```
{ NOINDENTGEN }
```

Notes

This command displays all row member names left-justified in the row names column without indenting members based on generation in the Database Outline. Indenting generations is generally not useful if you sort member names alphabetically by name in a report.

Default Value

By default, each generation is indented unless NOINDENTGEN is used.

See Also

- [INDENT](#)
- [INDENTGEN](#)

NOPAGEONDIMENSION

Turns off insertion of a new page when the member in the report from the same dimension as *member* changes in a row of the report.

Syntax

```
{ NOPAGEONDIMENSION mbrName }
```

Parameter Description

mbrName Single member whose dimension is part of the PAGEONDIMENSION declaration.

Notes

This command turns off insertion of a new page when the member in the report from the same dimension as *mbrName* changes in a row of the report. It is needed only after the PAGEONDIMENSION command has been used.

Example

{NOPAGEONDIMENSION Year} prevents a new page from being inserted when a member in the dimension Year changes, after PAGEONDIMENSION Year has been set.

See Also

- [NOSKIPONDIMENSION](#)
- [PAGEONDIMENSION](#)
- [SKIPONDIMENSION](#)

NOROWREPEAT

Prevents row member names from being repeated on each line of the report if the row member name does not change on the next line. This is the default.

Syntax

```
{ NOROWREPEAT }
```

Notes

This command prevents row member names from being repeated on each line of the report if the row member name does not change on the next line. NOROWREPEAT is only used to cancel the effects of the ROWREPEAT command. The ROWREPEAT command causes all row member names to be displayed on every line of the report even if the names for some members are the same.

Default Value

NOROWREPEAT is the default; you need only use this command after using ROWREPEAT.

Example

The following example is based on the Sample Demo database.

The following report is an example of the default behavior for row names not repeating. The names only print when they change.

```
<PAGE (Market, Accounts)
Chicago Sales

      <COLUMN (Scenario)
      Actual

<ROW (Year, Product)
{ NOROWREPEAT }
<CHILDREN Qtr1
<CHILDREN Audio!
{ ROWREPEAT }
<CHILDREN Qtr2 !
```

Which produces the following report:

```
                Chicago Sales Actual

Jan                Stereo                923
                   Compact_Disc         1,120
                   Audio                 2,043
Feb                Stereo                834
                   Compact_Disc         1,050
```

	Audio	1,884
Mar	Stereo	834
	Compact_Disc	980
	Audio	1,814
Qtr1	Stereo	2,591
	Compact_Disc	3,150
	Audio	5,741

Chicago Sales Actual

Apr	Stereo	821
Apr	Compact_Disc	985
Apr	Audio	1,806
May	Stereo	821
May	Compact_Disc	1,014
May	Audio	1,835
Jun	Stereo	834
Jun	Compact_Disc	1,022
Jun	Audio	1,856
Qtr2	Stereo	2,476
Qtr2	Compact_Disc	3,021
Qtr2	Audio	5,497

See Also

- [ROWREPEAT](#)

NOSKIPONDIMENSION

Prevents insertion of a new line when a member from the same dimension as *mbrName* changes in a row of the report.

Syntax

```
{ NOSKIPONDIMENSION mbrName }
```

Parameter Description

mbrName Single member that defines a dimension for which to halt line-skipping.

Notes

This command turns off insertion of a new line when the member in the report from the same dimension as *mbrName* in the command changes in a row of the report.

This command is required only after the SKIPONDIMENSION command.

Example

```
{NOSKIPONDIMENSION Year}
```

prevents the insertion of a new line when a member in the dimension Year changes after an occurrence of SKIPONDIMENSION Year.

See Also

- [NOPAGEONDIMENSION](#)

- [PAGEONDIMENSION](#)
- [SKIPONDIMENSION](#)

NOUNAMEONDIM

Turns off underlining for the new member name when the member in the report from the same dimension as the specified member changes in a row of the report.

Syntax

```
{ NOUNAMEONDIM mbrName }
```

Parameter Description

mbrName Member whose dimension is part of the UNAMEONDIM declaration.

Notes

This command turns off underlining for a new row when the member in the report from the same dimension as *mbrName* changes. It is needed only after the UNAMEONDIM command has been used.

See Also

- [NOPAGEONDIMENSION](#)
- [NOSKIPONDIMENSION](#)
- [PAGEONDIMENSION](#)
- [SKIPONDIMENSION](#)
- [UNAMEONDIMENSION](#)

OFFCOLCALCS

Disables all column calculations within the report.

Syntax

```
{ OFFCOLCALCS }
```

Notes

This command disables all column calculations within the report, for example, those calculations set by CALCULATE COLUMN. The column(s) defined for the calculation(s) display the value #MISSING to indicate no value was calculated for the column. This command temporarily turns off the calculations but does not remove them.

Example

See the example for the [CALCULATE COLUMN](#) command.

See Also

- [CALCULATE COLUMN](#)
- [CLEARROWCALC](#)

- [CLEARALLROWCALC](#)
- [OFFROWCALCS](#)
- [ONCOLCALCS](#)
- [ONROWCALCS](#)
- [PRINTROW](#)
- [REMOVECOLCALCS](#)
- [SETROWOP](#)

OFFROWCALCS

Temporarily disables all row calculations.

Syntax

```
{ OFFROWCALCS }
```

Notes

This command temporarily disables all row calculations, for example, those calculations set by [CALCULATE ROW](#). Subsequent rows of data do not contribute to a calculated row with an active [SETROWOP](#) until [ONROWCALCS](#) is issued. Disabling the calculations does not reset the values of the rows to zero. Instead, rows of data in the report after the command are ignored in the calculations.

Example

See the examples for the [CALCULATE ROW](#) command.

See Also

- [CALCULATE ROW](#)
- [CLEARROWCALC](#)
- [CLEARALLROWCALC](#)
- [OFFCOLCALCS](#)
- [ONCOLCALCS](#)
- [ONROWCALCS](#)
- [PRINTROW](#)
- [REMOVECOLCALCS](#)
- [SETROWOP](#)

OFSAMEGEN

Adds to the report the members from the same dimension and generation as the specified member. This command does not apply to Hybrid Analysis members.

Syntax

```
<OFSAMEGEN mbrName
```

Parameter Description

mbrName Single member that designates the dimension and generation to retrieve.

Notes

Generations are counted starting at the top of the dimension. The top of the dimension is generation 1; its children are generation 2. Each child's generation number is one greater than its parent's.

Example

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual

<COLUMN (Year)
<CHILDREN Year

<ROW (Product)
<OFSAMEGEN VCR
!
```

This example produces the following report:

```
Chicago Sales Actual

      Qtr1   Qtr2   Qtr3   Qtr4   Year
=====
Stereo      2,591   2,476   2,567   3,035   10,669
Compact_Disc 3,150   3,021   3,032   3,974   13,177
Television   4,410   4,001   4,934   6,261   19,606
VCR          3,879   3,579   4,276   4,877   16,611
Camera       2,506   2,522   2,602   3,227   10,857
```

See Also

- [ALLINSAMEDIM](#)
- [CHILDREN](#)
- [DESCENDANTS](#)
- [ONSAMELEVELAS](#)

ONCOLCALCS

Re-enables column calculations in the report after they have been disabled by OFFCOLCALCS.

Syntax

```
{ ONCOLCALCS }
```

Notes

This command is required after the OFFCOLCALCS command, which disables column calculations.

Example

See the example for the [CALCULATE COLUMN](#) command.

See Also

- [CALCULATE COLUMN](#)
- [CLEARROWCALC](#)
- [CLEARALLROWCALC](#)
- [OFFCOLCALCS](#)
- [OFFROWCALCS](#)
- [ONROWCALCS](#)
- [PRINTROW](#)
- [REMOVECOLCALCS](#)
- [SETROWOP](#)

ONROWCALCS

Re-enables all row calculations after they have been disabled by OFFROWCALCS. Each subsequent row of data after using the command is calculated.

Syntax

```
{ ONROWCALCS }
```

Notes

This command is required after the OFFROWCALCS command, which disables the row calculation(s).

Example

See the example for the [CALCULATE ROW](#) command.

See Also

- [CALCULATE ROW](#)
- [CLEARROWCALC](#)
- [CLEARALLROWCALC](#)
- [OFFCOLCALCS](#)
- [ONCOLCALCS](#)
- [REMOVECOLCALCS](#)

ONSAMELEVELAS

Adds to the report all members on the same level as the specified member. This command does not apply to Hybrid Analysis members.

Syntax

```
<ONSAMELEVELAS mbrName
```


Parameter Description

mbrName Single member that designates the dimension and generation to retrieve.

Notes

Levels are counted up from the bottom of the dimension. Members in the database outline with no children are level 0; their parents are level 1, and so on. The level for a child is always 1 lower than its parent.

Example

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual

    <COLUMN (Year)
    <ICHILDREN Year

<ROW (Product)
<ONSAMELEVELAS Audio
    !
```

This example produces the following report:

```
                Chicago Sales Actual
           Qtr1   Qtr2   Qtr3   Qtr4   Year
    =====  =====  =====  =====  =====
Audio       5,741   5,497   5,599   7,009   23,846
Video      10,795  10,102  11,812  14,365  47,074
```

See Also

- [ALLINSAMEDIM](#)
- [CHILDREN](#)
- [DESCENDANTS](#)
- [OFSAMEGEN](#)

ORDER

Specifies the order of columns in a report, based on the original ordering of the columns.

Make sure you specify all the report columns in the ORDER command unless you use FIXCOLUMNS. ORDER simply moves the listed columns to locations in the final report but does not shift the unlisted columns to make room for the columns moved. If you have a five column report and you specify the command {ORDER 2 3 4}, you see columns 2, 3 and 4 in the report followed again by columns 3 and 4. If you really want a 3 column report, use {FIXCOLUMNS 3}.

Calculated data columns have column numbers which begin after the last regular data column. In other words, if each output data row had:

- 2 row names;

- 3 regular data columns; and
- 2 calculated data columns

then columns 0 and 1 are the row name column numbers; 2, 3, and 4 are the regular data column numbers; and 5 and 6 are the calculated-data column members.

Syntax

```
{ ORDER columnList }
```

Parameter Description

columnList Numeric designations of the columns to rearrange, separated by a space between each column number. Each column number represents the *initial* positions of each column (from 0 to *n* where *n* is the last column, counting names, data, and calculated columns, respectively). The position of each number in the *columnList* represents the new order in which you want the columns to be displayed.

Note: Using the ORDER command without a *columnList* resets the column order to the default setting (that is, 0, 1, 2, 3, 4, and so on).

Notes

- ORDER is a setting command.
- The first name column is designated as column 0. Column numbers then increment, starting with any additional row name columns, then the data columns, followed by calculated data columns.

Example

The following example is based on the Sample Basic database.

```
<PAGE (Measures, Market)
Texas Sales
{ORDER 0 1 4 2 5 3 6 BLOCKHEADERS}
  <COLUMN (Scenario, Year)
  Actual Budget
  Jan Feb Mar
<ROW (Product)
<DESCENDANTS "100"
  !
```

This script arranges the Jan, Feb, and Mar columns side-by-side.

	Sales Texas					
	Actual	Budget	Actual	Budget	Actual	Budget
	Jan	Jan	Feb	Feb	Mar	Mar
	=====	=====	=====	=====	=====	=====
100-10	452	560	465	580	467	580
100-20	190	230	190	230	193	240
100-30	#Missing	#Missing	#Missing	#Missing	#Missing	#Missing

See Also

- [FIXCOLUMNS](#)

- [NAMESCOL](#)

ORDERBY

Orders the rows in a report according to data values in the specified columns.

Syntax

```
<ORDERBY ( [<rowgroupDimension> ,] <column> [<direction>]{ ,<column> [<direction>] } )
```

Parameter	Description
<Optional rowgroup Dimension>	Row grouping dimension that determines the rows to sort as a set.
<column>	@DATACOL (<colnumber>) @DATACOL (<colnumber>) where <colnumber> is the target column number; must be between 1 and the maximum number of columns in the report.
<direction>	You can specify multiple columns with different sorting directions where: ASC is the ascending sort DESC is the descending sort

Notes

You can use ORDERBY, TOP, BOTTOM, and RESTRICT in the same report script, but you can use each command only once per report. If you repeat the same command in a second report in the same report script, the second command overwrites the first. Place global script formatting commands, for example, SAVEROW, before a PAGE, COLUMN command or associated member (for example, <CHILDREN or <IDESCENDANTS).

If any of the ORDERBY, TOP, BOTTOM, or RESTRICT commands exist together in a report script, the row group dimension <rowgroupDimension> should be the same. This restriction removes any confusion about the sorting and ordering of rows within a row group. Otherwise, an error is issued.

If TOP or BOTTOM commands exist in the same report with ORDERBY, the ordering column of ORDERBY need not be the same as that of TOP or BOTTOM.

The ORDERBY, TOP and BOTTOM commands sort a report output by its data values. The RESTRICT command restricts the number of valid rows for the report output. Their order of execution is:

1. Any sorting command that sorts on member names (for example <SORTDESC or <SORTASC)
2. RESTRICT
3. TOP and BOTTOM
4. ORDERBY

This order of execution applies irrespective of the order in which the commands appear in the report script.

For an example that uses TOP, BOTTOM, ORDERBY, and RESTRICT together, see the entry for the BOTTOM command.

Default Value

The innermost row grouping is the default row group dimension. Default direction is ascending.

Example

```
//Page dimension
<PAGE("Measures")

//Column dimensions
<COLUMN("Scenario", "Year")

//Row dimensions
<ROW("Market", "Product")

// Page Members
"Sales"

// Column Members
"Scenario"

"Jan" "Feb" "Mar"

// Row Members
"New York"

"Product" "100" "100-10" "100-20" "100-30" "200" "200-10" "200-20" "200-30" "200-40"
"300" "300-10" "300-20" "300-30" "400" "400-10" "400-20" "400-30" "Diet" "100-20"
"200-20" "300-30"

// Data sorting
<ORDERBY ("Product", @DATACOL(1) ASC, @DATACOL(2) DESC, @DATACOL(3) ASC)
!
// End of report
```

Which produces the following report based on the Sample Basic sample database:

		Sales Scenario		
		Jan	Feb	Mar
		=====	=====	=====
New York	100-20	#Missing	#Missing	#Missing
	100-30	#Missing	#Missing	#Missing
	200-20	#Missing	#Missing	#Missing
	200-30	#Missing	#Missing	#Missing
	300-30	#Missing	#Missing	#Missing
	Diet	#Missing	#Missing	#Missing
	200-10	61	61	63
	400-30	134	189	198
	300-20	180	180	182
	400-20	219	243	213

400-10	234	232	234
300-10	483	495	513
200-40	490	580	523
200	551	641	586
400	587	664	645
300	663	675	695
100-10	678	645	675
100	678	645	675
Product	2,479	2,625	2,601

See Also

- [RESTRICT](#)
- [TOP](#)
- [BOTTOM](#)

OUTALT

Sets the output alias to the database outline alias name, as defined in the current alias table.

Syntax

```
<OUTALT
```

Notes

- OUTALT cannot be used on duplicate member outlines. See [REPALIAS](#).
- OUTALT is used to reset the output alias to the Database Outline alias name. Use this command to restore the default alias after OUTALTMBR or OUTMBRALT have been used to redefine the alternate name.
- You must precede the OUTALT command with OUTALTNAMES to display the alias (rather than the member name).

Example

The following example is based on the Sample Basic database.

```
<PAGE (Product, Measures)
<COLUMN (Scenario, Year)
{OUTALTNAMES}
<OUTMBRALT
Actual
<CHILDREN Qtr1
<ROW Market)
<IDESCENDANTS "300"
<OUTALT
<IDESCENDANTS "300"
!
<OUTALT
<IDESCENDANTS "300"
!
```

This example produces the following report:

```
300-10 Measures Actual
```

	Jan	Feb	Mar
Market	800	864	880

Vanilla Cream Measures Actual

	Jan	Feb	Mar
Market	220	231	239

Diet Cream Measures Actual

	Jan	Feb	Mar
Market	897	902	896

Cream Soda Measures Actual

	Jan	Feb	Mar
Market	1,917	1,997	2,015

Dark Cream Measures Actual

	Jan	Feb	Mar
Market	800	864	880

Vanilla Cream Measures Actual

	Jan	Feb	Mar
Market	220	231	239

Diet Cream Measures Actual

	Jan	Feb	Mar
Market	897	902	896

Cream Soda Measures Actual

	Jan	Feb	Mar
Market	1,917	1,997	2,015

See Also

- [OUTALTMBR](#)
- [OUTALTNAMES](#)
- [OUTMBRALT](#)
- [OUTMBRNames](#)

OUTALTMBR

Sets the output alias to the database outline alias name (as defined in the current alias table) followed by the database outline member name.

Syntax

```
<OUTALTMBR
```

Notes

- Separate the alias and member name with a single space.
- To produce reports that display the alternate name for a member, you must also use the { OUTALT NAMES } command. If no alternate name exists, only the member name is displayed.
- OUTALT MBR cannot be used on duplicate member outlines. See [REP ALIAS MBR](#).

Example

The following example is based on Sample Basic.

```
<PAGE (Product, Measures)
<COLUMN (Scenario, Year)
{OUTALT NAMES}
<OUTALT MBR
Actual
<CHILDREN Qtr1
<ROW (Market)
<IDESCENDANTS "300"
!
```

This example produces the following report:

```
          300-10 Measures Actual
          Jan      Feb      Mar
          =====
Market    800      864      880

          Vanilla Cream 300-20 Measures Actual
          Jan      Feb      Mar
          =====
Market    220      231      239

          Diet Cream 300-30 Measures Actual
          Jan      Feb      Mar
          =====
Market    897      902      896

          Cream Soda 300 Measures Actual
          Jan      Feb      Mar
          =====
Market    1,917    1,997    2,015
```

See Also

- [OUTALT](#)
- [OUTALT NAMES](#)
- [OUTMBRALT](#)
- [REP ALIAS MBR](#)

OUTALT NAMES

Displays alias names for members in a report.

May be used in conjunction with OUTMBRNAME to switch between member names and alias names in report rows.

The member name, not the alias name, is the default for reporting.

Syntax

```
{ OUTALTNAMES }
```

Notes

- OUTALTNAMES cannot be used on duplicate member outlines. See [REPALIAS](#).
- OUTALTNAMES is a setting command.
- The OUTALTMBR or OUTMBRALT commands may be used to redefine the alternate names definition.

Example

The following example is based on Sample Basic.

```
{WIDTH 15}
//{OUTALTNAMES} If used (commented out), displays alias names for column headers
<PAGE (Measures)
Sales
<COL (Year, Market, Scenario)
Jan Feb Mar
  East Actual
<ROW(Measures)
{OUTALTNAMES}
// These members display with aliases.
<IDESCENDANTS "100"
{OUTMBRNames}
// These members display their member names as defined in the outline.
<IDESCENDANTS "200"
{OUTALTNAMES}
// Switches back to alias names, as defined in the current alias table.
<IDESCENDANTS "400"
!
```

This example produces the following report:

	Sales East Actual		
	Jan	Feb	Mar
	=====	=====	=====
Cola	1,812	1,754	1,805
Diet Cola	200	206	214
Caffeine Free Cola	93	101	107
Colas	2,105	2,061	2,126
200-10	647	668	672
200-20	310	310	312
200-30	#Missing	#Missing	#Missing
200-40	896	988	923
200	1,853	1,966	1,907
Grape	562	560	560
Orange	219	243	213
Strawberry	432	469	477
Fruit Soda	1,213	1,272	1,250

See Also

- [OUTALT](#)
- [OUTALTMBR](#)
- [OUTMBRALT](#)
- [OUTMBRNames](#)

OUTALTSELECT

Selects an alias table in a report script.

The table remains in effect until another <OUTALTSELECT command executes. This lets you use different alias tables for different dimensions in a report script.

Syntax

```
<OUTALTSELECT AliasTableName
```

Parameter	Description
-----------	-------------

AliasTableName	The name of the selected alias table associated with the database outline.
----------------	----------------------------------------------------------------------------

Notes

- OUTALTSELECT can be used on unique member outlines or duplicate member outlines.

Example

The following example is based on Sample Basic, using two different alias tables: Long Names and Default.

```
<PAGE("Scenario")
<COLUMN("Year", "Market")
<ROW("Measures", "Product")
<LINK( <CHILDREN("Qtr4"))
<LINK( <CHILDREN("South"))
<OUTALTSELECT "Long Names"
{OUTALTNAMES}"100-10"
"100-20"
"100-30"
<OUTALTSELECT Default
{OUTALTNAMES}
"200-10"
"200-20"
"200-30"
!
```

See Also

- [REPALIAS](#)
- [REPALIASMBR](#)
- [REPMBR](#)
- [REPMBRALIAS](#)
- [OUTALTMBR](#)
- [OUTALTNAMES](#)

- [OUTMBRALT](#)
- [OUTMBRNames](#)

OUTFORMATTEDMISSING

Formats missing values in reports instead of the missing alias. By default, missing values are not formatted. Only cells with non-numeric type are formatted.

Syntax

```
{ OUTFORMATTEDMISSING }
```

Parameter Description

Example

See Also

- [WITHATTR](#)

OUTFORMATTEDVALUES

Generates formatted cell values in the report instead of cell values. By default cell values are reported. Cells with missing values will not be formatted.

Syntax

```
{ OUTFORMATTEDVALUES }
```

Parameter Description

Example

See Also

- [WITHATTR](#)

OUTMBRALT

Sets the output name to the database outline member name followed by the outline alias, as defined in the current alias table.

The member name and alias are separated by a single space.

Syntax

```
<OUTMBRALT
```

Notes

- OUTMBRALT cannot be used on duplicate member outlines. See [REPMBRALIAS](#).
- You must precede the OUTMBRALT command with OUTALTNAMES to display the alias, followed by the member name (rather than the member name alone).
- OUTMBRALT cannot be used on duplicate member name outlines.
- REPMBRALIAS can be used on both unique and duplicate member name outlines. REPMBRALIAS supercedes OUTMBRALT.

Example

The following example is based on Sample Basic.

```
<PAGE (Product, Measures)
<COLUMN (Scenario, Year)
{OUTALTNAMES}
<OUTMBRALT
Actual
<CHILDREN Qtr1
<ROW (Market)
<IDESCENDANTS "300"
!
```

This example produces the following report:

```
          300-10 Measures Actual
          Jan      Feb      Mar
          =====
Market    800      864      880

          300-20 Vanilla Cream Measures Actual
          Jan      Feb      Mar
          =====
Market    220      231      239

          300-30 Diet Cream Measures Actual
          Jan      Feb      Mar
          =====
Market    897      902      896

          300 Cream Soda Measures Actual
          Jan      Feb      Mar
          =====
Market    1,917    1,997    2,015
```

See Also

- [OUTALT](#)
- [OUTALTMBR](#)
- [OUTALTNAMES](#)
- [OUTMBRNAMES](#)
- [REPMBRALIAS](#)

OUTMBRNAMES

Reverts to the default member name display after the OUTALTNAMES command has been used to display alternate names.

The member name is the default for reporting.

Syntax

```
{ OUTMBRNAMES }
```

Notes

- OUTMBRNAMES cannot be used on duplicate member outlines. See [REPMBR](#).

See Also

- [OUTALT](#)
- [OUTALTMBR](#)
- [OUTALTNAMES](#)
- [OUTMBRALT](#)

OUTMEANINGLESS

Displays #ME in reports for cells that are meaningless because no base member-attribute member combination exists.

Syntax

```
{ OUTMEANINGLESS }
```

Parameter Description

Example

See Also

- [WITHATTR](#)

OUTPUT

Resumes output, reversing the action of SUPOUTPUT.

Syntax

```
{ OUTPUT }
```

Notes

This command causes Report Writer to resume output with the member specifications in effect when the OUTPUT command was issued. It will not "remember" where it was when the SUPOUTPUT command was issued. Further, any formatting commands that were issued in the

interim will also be in effect. Thus, you can use the SUPOUTPUT command to suppress all output from a portion of the report script.

See Also

- [SUOUTPUT](#)

OUTPUTMEMBERKEY

Displays a member identifier (in addition to the member or alias name) for any duplicate member names. OUTPUTMEMBERKEY applies to duplicate member outlines only.

Syntax

<OUTPUTMEMBERKEY

Notes

- OUTPUTMEMBERKEY is primarily for use in programing applications.
- OUTPUTMEMBERKEY cannot be used in combination with the existing commands OUTMBRALT, OUTALTMBR, OUTALT, OUTALTNames, OR OUTMBRNames.
- [SORTMBRNames](#) does not sort by member identifier.

See Also

- [REPQUALMBR](#)
- [REPMBR](#)
- [REPALIAS](#)
- [REPMBRALIAS](#)
- [REPALIASMBR](#)

PAGE

Defines which dimensions are displayed as page members in the final report.

This command specifies the dimension or dimensions to be used such that each member or combination of members of these dimensions is an attribute of all data cells on a page.

Page members are displayed at the top of the report above the column members. Any member in the report specification from the same dimension as a member in the PAGE command is a page member. Only one member at a time from each page dimension is displayed in the page heading at the top of each page.

Each time any member from one of the dimensions in the page heading changes, it creates a new page heading. The order of the dimensions in the PAGE command determines the order in which members occur in the page heading. The member from the first dimension is displayed first, followed by the second and so on.

On any single report page, the current page members are representative of (are attributes of) all the data cells on the page.

Syntax

```
<PAGE ( dimList )
```

Parameter Description

dimList Dimension name or a comma-delimited list of dimensions.

Notes

- If dimension names contain spaces or consist of numbers, they must be enclosed in double quotes.
- Essbase automatically generates new page headings when dimensions change. Essbase does not, however, automatically generate page breaks. To specify page breaks when dimensions change, use the PAGEONDIMENSION format command, described in the Data Formatting Commands section later in this chapter.
- When more than one dimension is specified, the last dimension in the list changes most frequently. For example, <PAGE (Measures, Market) lists all values for Sales East (New York, Massachusetts, Florida, etc.), then lists all values for Sales West. After all Markets have been cycled, the next Measure will replace Sales, and then Markets will cycle through again.

Example

```
<PAGE (Measures, Market)
```

Creates a report based on member combinations of dimensions Measures and Market. The first page of the report lists all values for Sales, East; the next page lists all values for Sales, West; When all children of Market have been extracted, the report continues with Cost of Goods Sold, East followed by Cost of Goods Sold, West, and so on.

See Also

- [COLUMN](#)
- [ROW](#)

PAGEHEADING

Displays the page heading before the next data-output row.

Otherwise, a new page heading occurs only if the page or column members change, a page is generated (for example, page length is exceeded or a NEWPAGE command is issued), or a page header has not been done for this page and the first output row on the page is ready to print.

If PAGEHEADING is specified between the STARTHEADING and ENDHEADING commands, however, the page heading is displayed with the heading and not immediately. This command also permanently nullifies the effect of a previously issued SUPPAGEHEADING command.

The page heading is the default heading, which contains the current page members.

Syntax

```
{ PAGEHEADING }
```

Notes

- The TEXT and SUPPRESSHEADING command can be used to customize page heading text and placement.
- By default, page and column headers (together called the HEADING) are turned on. This means they are displayed prior to the first actual output row in a report, and are reset to display again whenever:
 1. A new page is generated.
 2. Any member in the page or column dimensions changes.
 3. A specific COLHEADING, PAGEHEADING, or IMMHEADING dictates a new heading. Once they are reset to display, they are output just prior to the new non-suppressed output row.
- IMMHEADING produces a new page and column heading immediately, without waiting for the next non-suppressed output line.

Example

The PAGEHEADING command in the following report inserts the page heading members in the report for a second time.

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual
    <COLUMN (Year)
        <CHILDREN (Year)
<ROW (Product)
Television
VCR
{ SKIP PAGEHEADING SKIP }
Compact_Disc
Stereo
    !
```

This example produces the following report:

```
                Chicago Sales Actual
                Qtr1   Qtr2   Qtr3   Qtr4   Year
                =====
Television      4,410  4,001  4,934  6,261  19,606
VCR             3,879  3,579  4,276  4,877  16,611
```

```
                Chicago Sales Actual
Compact_Disc    3,150  3,021  3,032  3,974  13,177
Stereo         2,591  2,476  2,567  3,035  10,669
```

See Also

- [COLHEADING](#)
- [HEADING](#)
- [PAGE](#)
- [SUPALL](#)
- [SUPCOLHEADING](#)

- [SUPHEADING](#)
- [SUPPAGEHEADING](#)
- [TEXT](#)

PAGELength

Sets the maximum number of lines for one page in the report.

Syntax

```
{ PAGELength [ lines ] }
```

Parameter Description

lines	Optional total number of output lines for the size of paper you are using. Because the Report Writer does not recognize any of the font characteristics of the output report, it operates based on lines rather than inches.
-------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Notes

Default Value

The defaults are FEEDON and a PAGELength of 66 lines, which normally translates to an 11-inch-long page. This value is assumed if *lines* is not given.

This command sets the maximum number of lines for one page in the report. After displaying the number of lines, a page break is inserted, followed by the heading. The page break is not inserted if a SUPFEED command has been used. The heading is displayed at the start of the new page unless SUPHEADING has been used.

If you are using legal size paper, the value should be 84 lines. If you are using A4 paper, the value should be 70 lines.

Example

{ PAGELength 50 } sets the maximum number of lines for one page to 50.

See Also

- [LMARGIN](#)
- [WIDTH](#)

PAGEONDimension

Performs a page break whenever a member from the same dimension as the specified member changes from one line in the report to the next.

Syntax

```
{ PAGEONDimension mbrName }
```


Parameter Description

mbrName Single member. If any member of the same dimension increments, a new page is started.

Notes

This command performs a page break whenever a member from the same dimension as the member in the command changes from one line in the report to the next.

With the ROW command, you can display members from several dimensions in columns on the side of the report. At least one member changes from one of these dimensions for each row of the report.

PAGEONDIMENSION causes a new page to begin when the member from the selected dimension changes. A single report can have several PAGEONDIMENSION commands to page on different dimensions which change.

When combined with UNAMEONDIMENSION and SKIPONDIMENSION, UNAMEONDIMENSION is processed first followed by SKIPONDIMENSION and PAGEONDIMENSION in order.

Example

The command { PAGEONDIMENSION Year } inserts a page break before displaying the members Qtr2, Qtr3, and Qtr4 in the following report below. On each new page, the heading members Chicago, Sales and Actual are displayed at the top of the page.

```
<PAGE (Market, Accounts)
Chicago Sales Actual

      <COLUMN (Scenario)
      <CHILDREN Year

<ROW (Year, Product)
{ PAGEONDIMENSION Year }
<ICHILDREN Audio
!
```

This example produces the following report:

```
Chicago Sales Actual

Qtr1   Stereo      2,591
       Compact_Disc 3,150
       Audio       5,741

Chicago Sales Actual

Qtr2   Stereo      2,476
       Compact_Disc 3,021
       Audio       5,497

Chicago Sales Actual

Qtr3   Stereo      2,567
```

Compact_Disc	3,032
Audio	5,599

Chicago Sales Actual

Qtr4	Stereo	3,035
	Compact_Disc	3,974
	Audio	7,009

See Also

- [NOPAGEONDIMENSION](#)
- [NOSKIPONDIMENSION](#)
- [SKIPONDIMENSION](#)

PARENT

Adds the parent of the member to the report.

Syntax

```
<PARENT mbrName
```

Parameter Description

mbrName Single member, which must not be the dimension (top) member.

Example

```
<PARENT Jan
```

adds Qtr1 to the report.

See Also

- [ANCESTORS](#)
- [CHILDREN](#)
- [DESCENDANTS](#)

PERSPECTIVE

Sets the perspective, a tuple or REALITY, for a varying attribute dimension for a report.

Syntax

```
<PERSPECTIVE(tuple, attrDim)
```

Parameter Description

tuple	(m1, m2, ..., mX) REALITY This is the perspective tuple to be applied for the given attribute dimension. <ul style="list-style-type: none">(m1, m2, ..., mN) Level-0 members from one or more independent dimensions for <code>attrDim</code> may be part of the input tuple.REALITY The REALITY keyword indicates using independent members from the current query-calculation context. When explicit perspectives are missing for an attribute dimension, the default usage for the perspective is REALITY.
attrdim	The varying attribute dimension to which the perspective applies. May be any member from attribute dimension hierarchy.

Notes

- Without the use of the perspective command, the default perspective will be used.
- The perspective specified for an attribute dimension influences the attribute calculations in the query. The following Report Writer commands involving attributes honor the prevailing perspective:
 - <Attribute `attMbrName`
 - <WithAttr(`dimName`, "operator", `value`)
- Only the first the perspective command in a report is honored. Any other perspective commands are ignored.

Example

```
<PERSPECTIVE((Jan), Ounces)
```

```
<PERSPECTIVE((Jan, California), Ounces)
```

See Also

- [WITHATTREX](#)
- [ATTRIBUTEVA](#)

PRINTROW

Displays the calculated *rowName* with its current values.

Syntax

```
{ PRINTROW "rowName" }
```

Parameter Description

"rowName" Character string, enclosed by quotation marks, which designates a previously declared calculated row. When the command is issued, the designated row is printed immediately in the report.

Example

See the examples for the [CALCULATE COLUMN](#) command.

See Also

- [CALCULATE COLUMN](#)
- [CLEARROWCALC](#)
- [CLEARALLROWCALC](#)
- [OFFCOLCALCS](#)
- [OFFROWCALCS](#)
- [ONCOLCALCS](#)
- [ONROWCALCS](#)
- [REMOVECOLCALCS](#)
- [RENAME](#)
- [SAVEANDOUTPUT](#)
- [SAVEROW](#)
- [SETROWOP](#)

PYRAMIDHEADERS

Displays column members in centered, pyramid-shaped levels above columns (the default style used by symmetric reports).

Syntax

```
{ PYRAMIDHEADERS }
```

Notes

This command displays column members in centered, pyramid-shaped levels over the columns in the report. Pyramid display of column members is the default method for displaying column members.

Pyramid headers cannot be used with asymmetric reports unless the report is extracted as a symmetric report and reordered or truncated to make it asymmetric.

Default Value

Default for symmetric reports. Also resets the default column display following a [BLOCKHEADERS](#) command.

Example

The following example is based on Sample Basic.

```
<PAGE (Measures, Market)
Sales
{WIDTH 7}
{ BLOCKHEADERS }
  <COLUMN (Scenario, Year)
    Actual Budget
    Jan Feb Mar
<ROW (Market)
```

```

<CHILD "200"
!
{PYRAMIDHEADERS}
<CHILD "300"
!

```

This example produces the following report:

	Sales			Market		
	Actual	Actual	Actual	Budget	Budget	Budget
	Jan	Feb	Mar	Jan	Feb	Mar
	=====	=====	=====	=====	=====	=====
200-10	3,220	3,348	3,326	3,230	3,370	3,370
200-20	3,122	3,161	3,203	3,090	3,120	3,190
200-30	1,478	1,463	1,499	1,310	1,290	1,330
200-40	896	988	923	870	950	890

	Sales			Market		
	Actual			Budget		
	Jan	Feb	Mar	Jan	Feb	Mar
	=====	=====	=====	=====	=====	=====
300-10	3,517	3,613	3,650	2,950	3,050	3,080
300-20	1,397	1,417	1,434	1,140	1,160	1,170
300-30	2,960	3,016	2,993	2,560	2,590	2,580

See Also

- [BLOCKHEADERS](#)

QUOTEMBRNAMES

Displays all the member names within quotation marks in the report script output when run through interfaces such as Administration Services, ESSCMD, and MaxL. Note that when the report script is run through the Spreadsheet Add-in or GRID API, the members are not returned within quotation marks.

Syntax

```
<QUOTEMBRNAMES
```

Notes

QUOTEMBRNAMES can occur anywhere in a report script. This command is useful when using the Report Writer to export data intended for reloading a database without the use of a data load rule file.

Note: When used in a report script that also uses the RENAME report command, names substituted using the RENAME command are not enclosed in quotation marks.

Example

```

<PAGE (Scenario)
<COLUMN (Year)
<ROW (Product, Market, Measures)

```

```
<QUOTE MBRNAMES  
{ ROWREPEAT }  
  
<CHILDREN Year  
<DIMBOTTOM Product  
<DIMBOTTOM Market  
<CHILDREN Profit  
!
```

REMOVECOLCALCS

Removes all column calculation definitions from the report.

Syntax

```
{ REMOVECOLCALCS }
```

Notes

This command removes all column calculation definitions from the report. The data values for any calculated columns are no longer calculated or displayed. This may be used if the limit of declared column calcs (50) is a problem. If the previous column calcs are no longer needed, they can be freed, creating room for up to 50 more.

See Also

- [CALCULATE COLUMN](#)
- [CLEARROWCALC](#)
- [CLEARALLROWCALC](#)
- [OFFCOLCALCS](#)
- [OFFROWCALCS](#)
- [ONCOLCALCS](#)
- [ONROWCALCS](#)
- [PRINTROW](#)
- [SETROWOP](#)

RENAME

Renames a member within the report.

Syntax

```
{ RENAME "newMbrName" } mbrName
```

Parameter	Description
-----------	-------------

"newMbrName"	Valid member name, enclosed in quotation marks, to be used as the replacement name.
--------------	-------------------------------------------------------------------------------------

mbrName	Name of the member that you want to rename temporarily.
---------	---------------------------------------------------------

Notes

This command renames a member within the report. This is a way of creating a temporary alias that applies to a single member, and it applies only within the report. Note that when you assign a temporary name to a member name, you do not have to state the member name again before or on the following line after the RENAME command. However, if you do state the member name later in the report, but not immediately on the next line after the RENAME command, the temporary name will be reset to its original member name.

Example

```
{RENAME "Video"} Visual
```

renames the Visual member to "Video" in the report.

REPALIAS

Displays alias names for members of the dimension specified.

If no alias exists for a member, the member name only is displayed. The current alias table is used unless [OUTALTSELECT](#) is used to specify an alternative alias table.

Syntax

```
<REPALIAS dimensionname
```

Notes

- <REPALIAS "" specifies the command for all dimensions.
- REPALIAS can be used on unique member outlines or duplicate member outlines.
- Some formatting commands (for example, [RENAME](#)) do not work with REPALIAS.
- REPALIAS cannot be used in combination with the existing commands OUTMBRALT, OUTALTMBR, OUTALT, OUTALT NAMES, OR OUTMBR NAMES.

Example

The following example is based on Sample Basic.

```
{WIDTH 15}
<PAGE (Measures)
Sales
<COL (Year, Market, Scenario)
Jan Feb Mar
  East Actual
<ROW(Product)
<IDESCENDANTS "100"
<IDESCENDANTS "200"
<IDESCENDANTS "400"
<REPALIAS product
// Displays aliases for all Product members
!
```

This example produces the following report:

Sales East Actual

	Jan	Feb	Mar
	=====	=====	=====
Cola	1,812	1,754	1,805
Diet Cola	200	206	214
Caffeine Free Cola	93	101	107
Colas	2,105	2,061	2,126
Old Fashioned	647	668	672
Diet Root Beer	310	310	312
Sasparilla	#Missing	#Missing	#Missing
Birch Beer	896	988	923
Root Beer	1,853	1,966	1,907
Grape	562	560	560
Orange	219	243	213
Strawberry	432	469	477
Fruit Soda	1,213	1,272	1,250

See Also

- [OUTALTSELECT](#)
- [OUTPUTMEMBERKEY](#)
- [REPALIASMBR](#)
- [REPMBR](#)
- [REPMBRALIAS](#)
- [REPQUALMBR](#)

REPALIASMBR

Displays alias names followed by member names for members of the dimension specified in the report output.

The alias and member name are separated by a single space. If no alias exists for a member, the member name only is displayed. The current alias table is used unless [OUTALTSELECT](#) is used to specify an alternative alias table.

Syntax

```
<REPALIASMBR dimensionname
```

Notes

- <REPALIASMBR "" specifies the command for all dimensions.
- REPALIASMBR can be used on unique member outlines or duplicate member outlines.
- Some formatting commands (for example, [RENAME](#)) do not work with REPALIASMBR.
- REPALIASMBR cannot be used in combination with the existing commands OUTMBRALT, OUTALTMBR, OUTALT, OUTALT NAMES, OR OUTMBR NAMES.

Example

The following example is based on Sample Basic.


```

<PAGE (Product, Measures)
<COLUMN (Scenario, Year)
<REPALIASMBR Product
Actual
<CHILDREN Qtr1
<ROW (Market)
<IDESCENDANTS "300"
!
```

This example produces the following report:

```

Dark Cream 300-10 Measures Actual
      Jan      Feb      Mar
=====
Market      800      864      880

Vanilla Cream 300-20 Measures Actual
      Jan      Feb      Mar
=====
Market      220      231      239

Diet Cream 300-30 Measures Actual
      Jan      Feb      Mar
=====
Market      897      902      896

Cream Soda 300 Measures Actual
      Jan      Feb      Mar
=====
Market      1,917    1,997    2,015
```

See Also

- [OUTALTSELECT](#)
- [OUTPUTMEMBERKEY](#)
- [REPALIAS](#)
- [REPMBR](#)
- [REPMBRALIAS](#)
- [REPQUALMBR](#)

REPMBR

Displays member names only for members of the dimension specified.

Used with the commands [REPALIAS](#), [REPMBRALIAS](#), and [REPALIASMBR](#).

Syntax

<REPMBR *dimensionname*

Notes

- <REPMBR "" specifies the command for all dimensions.
- REPMBR can be used on unique member outlines or duplicate member outlines.
- Some formatting commands (for example, [RENAME](#)) do not work with REPMBR.
- REPMBR cannot be used in combination with the existing commands OUTMBRALT, OUTALTMBR, OUTALT, OUTALTNAMES, OR OUTMBRNames.

Example

The following example is based on Sample Basic.

```
<PAGE (Product, Measures)
<COLUMN (Scenario, Year)
//Displays aliases for all dimensions except the Product dimension. Displays member
names for the Product dimension.
<REPALIAS ""
<REPMBR Product
Actual
<CHILDREN Qtr1
<ROW (Market)
<IDESCENDANTS "300"
!
```

This example produces the following report:

```
300-10 Measures Actual
      Jan      Feb      Mar
=====
Market      800      864      880

300-20 Measures Actual
      Jan      Feb      Mar
=====
Market      220      231      239

300-30 Measures Actual
      Jan      Feb      Mar
=====
Market      897      902      896

300 Measures Actual
      Jan      Feb      Mar
=====
```

Market 1,917 1,997 2,015

See Also

- [OUTPUTMEMBERKEY](#)
- [REPALIAS](#)
- [REPALIASMBR](#)
- [REPMBRALIAS](#)
- [REPQUALMBR](#)

REPMBRALIAS

Displays member names followed by aliases for members of the dimension specified. The member name and alias are separated by a single space. If no alias exists for a member, the member name only is displayed. The current alias table is used unless [OUTALTSELECT](#) is used to specify an alternative alias table.

Syntax

```
<REPMBRALIAS dimensionname
```

Notes

- `<REPMBRALIAS ""` specifies the command for all dimensions.
- `REPMBRALIAS` can be used on unique member outlines or duplicate member outlines.
- Some formatting commands (for example, [RENAME](#)) do not work with `REPMBRALIAS`.
- `REPMBRALIAS` cannot be used in combination with the existing commands `OUTMBRALT`, `OUTALTMBR`, `OUTALT`, `OUTALTNAMES`, or `OUTMBRNames`.

Example

The following example is based on Sample Basic.

```
<PAGE (Product, Measures)
<COLUMN (Scenario, Year)
<REPMBRALIAS Product
Actual
<CHILDREN Qtr1
<ROW (Market)
<IDESCENDANTS "300"
!
```

This example produces the following report:

```
300-10 Dark Cream Measures Actual
      Jan      Feb      Mar
=====
Market      800      864      880

300-20 Vanilla Cream Measures Actual
      Jan      Feb      Mar
```

```

=====
Market          220      231      239

300-30 Diet Cream Measures Actual

          Jan      Feb      Mar
=====
Market          897      902      896

300 Cream Soda Measures Actual

          Jan      Feb      Mar
=====

Market          1,917    1,997    2,015

```

See Also

- [OUTALTSELECT](#)
- [OUTPUTMEMBERKEY](#)
- [REPALIAS](#)
- [REPALIASMBR](#)
- [REPMBR](#)
- [REPQUALMBR](#)

REPQUALMBR

Displays member names for any unique member names and a system generated identifier (for example, a qualified name) for any duplicate member names for the dimension specified. REPQUALMBR applies to duplicate member outlines only.

Syntax

<REPQUALMBR *dimensionname*

Notes

- <REPQUALMBR "" specifies the command for all dimensions.
- Some formatting commands (for example, [RENAME](#)) do not work with REPQUALMBR.
- REPQUALMBR cannot be used in combination with the existing commands OUTMBRALT, OUTALTMBR, OUTALT, OUTALTNAMES, OR OUTMBRNAMES.

See Also

- [OUTPUTMEMBERKEY](#)
- [REPALIAS](#)
- [REPALIASMBR](#)
- [REPMBR](#)
- [REPMBRALIAS](#)

RESTRICT

The RESTRICT command specifies the conditions that the row must satisfy before it becomes part of a result set.

Syntax

```
<RESTRICT (<column | value> <operator> <column | value>{<logicalOperator><column | value> <operator> <column | value>} )
```

Parameter	Description
<column >	@DATACOL (<colnumber>) @DATACOL (<colnumber>) where <colnumber> is the target column number; must be between 1 and the maximum number of columns in the report.
<value>	Cell data type (real number) #MISSING
<operator>	>, >= greater than, greater or equal <, <= less than, less than or equal = equal !&, <> not equal
<logicalOperator>	Report Writer processes logical operations from left to right without exception. Parentheses are not supported. The supported logical operators are AND and OR.

Notes

Restrictions set by this command are processed from left to right.

You can use only one RESTRICT command per report, with a maximum of nine operators included in the command. RESTRICT persists to the end of the report script unless overwritten. You can use RESTRICT, TOP, BOTTOM, and ORDERBY in the same report script, but you can use each command only once per report. If you repeat the same command in a second report in the same report script, the second command overwrites the first. Place global script formatting commands, for example, SAVEROW, before a PAGE, COLUMN command or associated member (for example, <CHILDREN or <IDESCENDANTS).

The RESTRICT command can appear anywhere in a script. If sorting commands, including TOP, BOTTOM, or ORDERBY occur in the same report, the order of execution is:

1. Any sorting command that sorts on member names (for example <SORTDESC or <SORTASC)
2. RESTRICT
3. TOP and BOTTOM
4. ORDERBY

This order of execution applies irrespective of the order in which the commands appear in the report script.

For an example that uses TOP, BOTTOM, ORDERBY, and RESTRICT together, see the entry for the BOTTOM command.

You can use configurable variables to specify the size of the internal buffers used for storing and sorting the extracted data. The following settings affect the way the RESTRICT, TOP, and BOTTOM commands work:

- Retrieval Buffer Size (a database setting)
- Retrieval Sort Buffer Size (a database setting)
- “NUMERICPRECISION” on page 477 (an essbase.cfg setting)

For more information on the database settings, see the *Oracle Essbase Database Administrator's Guide*.

Example

```
{ StartHeading
  SupPageHeading
  Skip
  Text C "Annual Report" 70 "*PageString"
  Skip
  Endheading }

// Display the rows where the value of column 3 is greater than 1,300
<RESTRICT (@DataCol(3) > +1300 )

// Page and column dimensions
<Page (Accounts, Scenario)
<Column (Scenario, Year)

// Scenario members
Actual Budget Scenario

// Row dimensions
<Row (Market, Product)

// Market members
<Ichildren Market

// Product members
<Idescendants Product

!
// End report
```

Which produces the following report based on the Demo Basic sample database:

Annual Report		Page: 1		
		Actual	Budget	Scenario
		=====	=====	=====
East	Compact_Disc	13,612	13,616	13,612
	Audio	13,438	14,551	13,438
	Television	11,911	14,780	11,911
	VCR	15,506	16,772	15,506
	Camera	5,721	7,079	5,721
	Visual	33,138	38,631	33,138

	Product	46,576	53,182	46,576
West	Compact_Disc	21,568	20,935	21,568
	Audio	22,488	22,308	22,488
	Television	10,688	13,535	10,688
	VCR	19,706	17,782	19,706
	Camera	9,957	12,397	9,957
	Visual	40,351	43,714	40,351
	Product	62,839	66,022	62,839
South	Television	5,278	9,395	5,278
	VCR	13,994	15,810	13,994
	Camera	5,293	7,220	5,293
	Visual	24,565	32,425	24,565
	Product	24,565	32,425	24,565
Market	Compact_Disc	35,180	34,551	35,180
	Audio	35,926	36,859	35,926
	Television	27,877	37,710	27,877
	VCR	49,206	50,364	49,206
	Camera	20,971	26,696	20,971
	Visual	98,054	114,770	98,054
	Product	133,980	151,629	133,980

See Also

- [TOP](#)
- [BOTTOM](#)
- [ORDERBY](#)

ROW

Determines the row dimensions for a report whose member names appear in the data rows of the report.

The member(s) in the command determine which dimensions from the Database Outline are displayed in the rows.

dimList is a list of members or dimension members that specifies the order, from left to right, in which the row headers are listed unless subsequently moved by ORDER or NAMESCOL. Each dimension may be represented only once in *dimList*.

Syntax

```
<ROW ( dimList )
```

Parameter Description

dimList Dimension name or a comma-delimited list of dimensions.

Notes

- If dimension names contain spaces or consist of numbers, they must be enclosed in double quotes.
- When more than one dimension is specified the first dimension in the list appears in the leftmost row Name column, the next dimension in the list appears nested to the right of the first, and so on.

- By default attribute calculation dimension members (for example, SUM, AVG) are displayed as columns. To display them in rows, you must include them in the ROW command.

Example

```
<ROW (Product)
```

creates a report with each member of Product as a row in the report.

See Also

- [COLUMN](#)
- [PAGE](#)

ROWREPEAT

Displays all applicable row members on each row of the report even if a member describing a row is the same as in the previous row.

Syntax

```
{ ROWREPEAT }
```

Notes

This command returns the report to displaying members that change from one line to the next.

Default Value

Default is NOROWREPEAT.

Example

The following example is based on Demo Basic.

The command { ROWREPEAT } causes the row member names Qtr1 through Qtr4 to repeat for each line showing Compact_Disc in the report where the duplications would normally be suppressed.

```
<PAGE Market, Accounts)
Chicago Sales
```

```
    <COLUMN Scenario)
    Actual Budget
```

```
<ROW Year, Product)
```

```
{ROWREPEAT}
```

```
<CHILDREN Year
<CHILDREN Audio
    !
```

This example produces the following report:

Chicago Sales

		Actual	Budget
		=====	=====
Qtr1	Stereo	2,591	2,800
Qtr1	Compact_Disc	3,150	3,050
Qtr2	Stereo	2,476	2,700
Qtr2	Compact_Disc	3,021	3,050
Qtr3	Stereo	2,567	2,750
Qtr3	Compact_Disc	3,032	3,050
Qtr4	Stereo	3,035	3,300
Qtr4	Compact_Disc	3,974	3,950

See Also

- [NOROWREPEAT](#)
- [ROW](#)

SAVEANDOUTPUT

Adds *rowMbr* to the report and creates a new calculated row whose default name is *rowMbr*, but which may be renamed with an optional name, *rowCalcName*, enclosed in quotation marks.

The command automatically stores the data associated with *rowMbr*, and this data can be referenced by `CALC ROW`, `CALC COLUMN`, `PRINTROW`, or any other command that can reference a calculated row.

When this command is used, the calculation operator for that command is set to OFF, so that its contents are not be affected unless the user explicitly turns the operator back on.

SAVEANDOUTPUT both captures data and outputs the result, whereas SAVEROW captures the output but suppress it.

Syntax

```
{ SAVEANDOUTPUT [ "rowCalcName" ] } rowMbr !
```

Parameter Description

"rowCalcName" Optional. Name, enclosed by quotation marks, for the calculated data row created by the SAVEROW command.

rowCalcName can be multi-part, separated by a tilde (~), as in the CALCULATE ROW and CALCULATE COLUMN syntax.

rowMbr Row member that determines the row name for the calculated data row.

Notes

A member and a calculated row can have the same name. Report Writer considers them separate entities even though they have the same name.

Example

The following example is based on Demo Basic.

```
{ TEXT 18 "Expenses as % of Sales for January" }
```

Jan Boston Audio

Actual Budget

```
{ SAVEANDOUTPUT } Sales !
```

```
{ CALCULATE COLUMN " Actual%" = 1 % "Sales" 1  
  CALCULATE COLUMN "Budget%" = 2 % "Sales" 2 }
```

```
COGS Misc  
Payroll  
Marketing  
      !
```

This example produces the following report:

Expenses as % of Sales for January

	Jan Boston Audio			
	Actual	Budget		
	=====	=====		
Sales	1,985	2,150		

	Jan Boston Audio			
	Actual	Budget	Actual%	Budget%
	=====	=====	=====	=====
Cost_of_Goods_Sold	941	1,007	47	47
Miscellaneous	4	0	0	0
Payroll	542	530	27	25
Marketing	134	130	7	6

See Also

- [CALCULATE COLUMN](#)
- [CALCULATE ROW](#)
- [CLEARROWCALC](#)
- [CLEARALLROWCALC](#)
- [OFFCOLCALCS](#)
- [OFFROWCALCS](#)
- [ONCOLCALCS](#)
- [ONROWCALCS](#)
- [OUTPUT](#)
- [PRINTROW](#)
- [REMOVECOLCALCS](#)
- [SAVEANDOUTPUT](#)
- [SAVEROW](#)
- [SUPOUTPUT](#)

SAVEROW

Creates a new calculated row whose default name is *rowMbr*, but which may be renamed with an optional name enclosed in quotation marks.

The command automatically stores the data associated with *rowMbr*, and this data can be referenced by any CALC ROW, CALC COLUMN, PRINTROW command, or any other that can reference a calculated row.

When the command is used, the calculation operator for that command is set to OFF, so that its contents are not affected unless the user explicitly turns the operator back on. SAVEROW captures the data, but suppresses its output.

Syntax

```
{ SAVEROW [ "newRowCalcName" ] } rowMbr !
```

Parameter	Description
newRowCalcName	Optional. Name, enclosed in quotation marks, for the data row created by the SAVEROW command. The name can be multi-part, separated by a tilde (~), as in the CALCULATE ROW and CALCULATE COLUMN syntax.
rowMbr	Default row member used to determine the row name for the calculated data row. <i>rowMbr</i> is the next member encountered after the { SAVEROW } command, so other intervening { } format commands or non-member-selecting < commands are allowed and do not affect which member is saved.

Notes

There is no conflict with a member and a calculated row having the same name. They are separate entities even though they have the same name.

Example

The following example is based on Demo Basic.

```
{TEXT 18 "Expenses as % of Sales for January"}
Jan Boston Audio

    Actual Budget

{SAVEROW} Sales !
{CALCULATE COLUMN " Actual%" = 1 % "Sales" 1
  CALCULATE COLUMN "Budget%" = 2 % "Sales" 2}
COGS Misc
Payroll
Marketing
Sales
    !
```

Which produces the following report:

```
Expenses as % of Sales for January

    Jan Boston Audio
```

	Actual	Budget	Actual%	Budget%
	=====	=====	=====	=====
Cost_of_Goods_Sold	941	1,007	47	47
Miscellaneous	4	0	0	0
Payroll	542	530	27	25
Marketing	134	130	7	6
Sales	1,985	2,150	100	100

See Also

- [SAVEANDOUTPUT](#)

SCALE

Scales the data in the report by multiplying it by a numeric value.

Syntax

```
{ SCALE factor [ columnList ] }
```

Parameter Description

factor Numeric value by which all output values are multiplied. The result is a scaled value.

columnList Optional. List of column numbers that this command affects.

Notes

This command affects only the columns specified in the command or all columns if none are specified. Stored data is not affected by this command.

Example

The command {SCALE .01} multiplies the data values in the second report by .01.

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual
  <COLUMN (Year)
    <CHILDREN Year
<ROW (Product)
<CHILDREN Audio
  !

{SCALE 2}
Chicago Sales Actual
  <CHILDREN Year
<CHILDREN Audio
  !
```

This example produces the following report:

```
Chicago Sales Actual

Qtr1   Qtr2   Qtr3   Qtr4
===== ===== ===== =====
Stereo  2,591  2,476  2,567  3,035
```

Compact_Disc 3,150 3,021 3,032 3,974

Chicago Sales Actual

	Qtr1	Qtr2	Qtr3	Qtr4
	=====	=====	=====	=====
Stereo	5,182	4,952	5,134	6,070
Compact_Disc	6,300	6,042	6,064	7,948

See Also

- [BRACKETS](#)
- [COMMAS](#)
- [DECIMAL](#)
- [SUPBRACKETS](#)
- [SUPCOMMAS](#)

SETCENTER

Sets a new centerline position on the page.

Syntax

```
{ SETCENTER charPosition }
```

Parameter	Description
-----------	-------------

<code>charPosition</code>	Integer representing a character position on your page. Character position is counted from the left edge of the page and is not affected by the left margin setting.
---------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------

Notes

This command sets a new centerline position on the page. Under normal circumstances, the center of the page is calculated based on the default page width and the left margin position until column members have been encountered, after which it defaults to the center of the data column area.

The SETCENTER command allows you to issue an arbitrary centerline position, which is then used for all centered text, including page headers. This can be helpful to center text before all the members defining the columns (and thus, the page width). It can also be used to reset the center in cases where the centering is not appealing when based on the exact center of the data columns.

SETROWOP

Defines on-the-fly calculations for a named row created with CALCULATE ROW.

This command determines the calculation for the calculated row specified in *rowCalcName*. The following table lists the operators you use for the *operation* in the command:

Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Percentages
OFF	Turns off the calculation

The addition operator, for example, sums all values in all rows output while the operation is on. The result in the calculated row may be printed with PRINTROW at any time. You may only use a single operator per calculated row. Before using the SETROWOP command, you must define the row name with the CALCULATE ROW command, or with SAVEROW or SAVEANDOUTPUT. Refer to the CALCULATE ROW command for more information on its ability to set the row operator.

If an *operation* is not specified, the default is + (add).

Syntax

```
{ SETROWOP "rowCalcName" [ operation ] }
```

Parameter	Description
-----------	-------------

rowCalcName	Named row, in double quotes, to which SETROWOP applies.
-------------	---------------------------------------------------------

operation	You can use any valid row calculation expression. SETROWOP accepts the same mathematical operators as CALCULATE ROW. In addition, SETROWOP accepts the OFF operator, which turns off row operations for rows that follow.
-----------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Notes

SETROWOP performs unary operations on the row or rows that follow. SETROWOP "*rowCalcName*" OFF turns off operations on subsequent rows.

Example

See the examples for CALCULATE ROW.

See Also

- [CALCULATE ROW](#)
- [CLEARROWCALC](#)
- [CLEARALLROWCALC](#)
- [OFFCOLCALCS](#)
- [OFFROWCALCS](#)
- [ONCOLCALCS](#)
- [ONROWCALCS](#)
- [OUTPUT](#)

- [PRINTROW](#)
- [REMOVECOLCALCS](#)
- [SAVEANDOUTPUT](#)
- [SAVEROW](#)
- [SUPOUTPUT](#)

SINGLECOLUMN

Displays a column heading when there is only one column member extracted in the report.

Syntax

```
<SINGLECOLUMN
```

Notes

This formatting command displays a column heading when there is only one column member selected in the report.

Example

```
<singlecolumn
{suppagehead}
<column(year)
<row(measures)
Profit Inventory Ratios
Qtr1
!
```

This examples produces the following report:

	Qtr1
	=====
Profit	24,703
Inventory	117,405
Ratios	55

See Also

- [COLHEADING](#)
- [PAGEHEADING](#)
- [SUPCOLHEADING](#)
- [SUPPAGEHEADING](#)

SKIP

Outputs a number of blank lines in the report or a single line if *n* is omitted from the command. The default value is single skip.

Syntax

```
{SKIP n }
```

Parameter Description

n Positive integer representing the number of lines to skip.

Notes

- SKIP is an output command.
- The value of *n* must be a positive integer.
- If you do not specify a value for *n*, {SKIP} defaults to 1.

Example

```
<PAGE (Measures, Market)
Texas Sales
    <COLUMN (Scenario, Year)
        Actual Budget
        Jan Feb
<ROW (Market)
<DESCENDANTS "100"
{SKIP 2}
<DESCENDANTS "200"
<DESCENDANTS "300"
!
```

Which inserts two blank lines between the rows containing descendants of member 100 and descendants of members 200 and 300.

See Also

- [NEWPAGE](#)
- [NOSKIPONDIMENSION](#)
- [SKIPONDIMENSION](#)

SKIPONDIMENSION

Inserts a blank line when a member from the same dimension as the specified member changes on the next line in the report.

Syntax

```
{ SKIPONDIMENSION mbrName }
```

Parameter Description

mbrName Name of single member. When a member from this dimension changes during report processing, a blank line is inserted before the member change.

Notes

This command outputs a blank line when a member from the same dimension as *mbrName* in the command changes on the next line in the report. With the ROW command, you can display members from several dimensions in columns on the side of the report. At least one member changes from one of these dimensions for each row of the report. The SKIPONDIMENSION

displays a blank line before the member from the dimension changes. When combined with UNAMEONDIMENSION and/or PAGEONDIMENSION, UNAMEONDIMENSION is processed first followed by SKIPONDIMENSION and PAGEONDIMENSION in order.

Example

The command {SKIPONDIMENSION Year} in the following report inserts a blank line before the row members Qtr2, Qtr3, and Qtr4 in the report.

```
<PAGE (Market, Accounts)
Chicago Sales
  <COLUMN (Scenario)
    Actual
  <ROW (Year, Product)
  { SKIPONDIMENSION Year }
<CHILDREN Year
<ICHILDREN Audio
  !
```

Chicago Sales Actual		
Qtr1	Stereo	2,591
	Compact_Disc	3,150
	Audio	5,741
Qtr2	Stereo	2,476
	Compact_Disc	3,021
	Audio	5,497
Qtr3	Stereo	2,567
	Compact_Disc	3,032
	Audio	5,599
Qtr4	Stereo	3,035
	Compact_Disc	3,974
	Audio	7,009

See Also

- [NOPAGEONDIMENSION](#)
- [NOSKIPONDIMENSION](#)
- [PAGEONDIMENSION](#)

SORTALTNAMES

Alphabetically sorts members by their alternate names within a member selection command (for example, <CHILDREN).

Syntax

```
<SORTALTNAMES
```

Notes

This command sorts alphabetically all members added with a member command (for example, <CHILDREN) by their alternate name. Members entered directly in the report specification without a member command, calculated rows and column names, or member commands encountered in the specification prior to the SORTALTNAMES command, are not affected by the command.

This command must precede the selection commands, for example, CHILDREN or DESCENDANTS. If no sorting commands are used, members are output in hierarchical order based on the member outline. Any sort command remains in effect until another sort command is issued.

Example

The following example is based on Demo Basic.

The command <SORTALTNAMES sorts the members added to the report with the <IDESCENDANTS Product command by the alternate name of each member. The command {OUTALTNAMES} causes alternate member names to be displayed in the report. {NOINDENTGEN} turns off hierarchical indenting so the row names line up. Indented row names are not particularly useful when the output is sorted on any criteria other than generation.

Chicago Sales Actual

	Qtr1	Qtr2	Qtr3	Qtr4
	=====	=====	=====	=====
Audio	5,741	5,497	5,599	7,009
Camera	2,506	2,522	2,602	3,227
Compact_Disc	3,150	3,021	3,032	3,974
Product	16,536	15,599	17,411	21,374
Stereo	2,591	2,476	2,567	3,035
Television	4,410	4,001	4,934	6,261
VCR	3,879	3,579	4,276	4,877
Visual	10,795	10,102	11,812	14,365

Chicago Sales Actual

	Qtr1	Qtr2	Qtr3	Qtr4
	=====	=====	=====	=====
Audio	5,740	5,375	5,509	6,488
CD	3,290	3,034	3,132	3,571
Camera	2,230	2,255	2,266	3,162
Items	15,812	15,050	16,716	19,159
Media	10,072	9,675	11,207	12,671
Radio	2,450	2,341	2,377	2,917
TV	4,197	3,757	4,740	5,000
Video	3,645	3,663	4,201	4,509

See Also

- [ALLINSAMEDIM](#)
- [CHILDREN](#)
- [DESCENDANTS](#)
- [SORTASC](#)

- [SORTDESC](#)
- [SORTGEN](#)
- [SORTLEVEL](#)
- [SORTMBRNames](#)
- [SORTNONE](#)

SORTASC

Specifies an ascending sort order.

Syntax

<SORTASC

Notes

This command determines the order in which members are sorted in member commands in the report specification. You use this command prior to the other sort commands including SORTALTNAMES, SORTGEN, SORTLEVEL and SORTMBRNames. With the SORTASC command, all following members selected are sorted into ascending order starting with either the letter "a" or the lowest generation and moving toward the letter "z" or the highest generation. Sorting in ascending order is the default sort order and is only changed with the SORTDESC command.

This command must precede the selection commands, or example, CHILDREN or DESCENDANTS. If no sorting commands are used, members are output in hierarchical order based on the member outline. Any sort command remains in effect until reset by another sort command.

The SORTASC command can be used to restore the default (ascending) sort order. It reverses the effects of a previously-specified SORTDESC command.

See Also

- [ALLINSAMEDIM](#)
- [CHILDREN](#)
- [DESCENDANTS](#)
- [SORTALTNAMES](#)
- [SORTDESC](#)
- [SORTGEN](#)
- [SORTLEVEL](#)
- [SORTMBRNames](#)
- [SORTNONE](#)

SORTDESC

Specifies a descending, hierarchical sort order.

Syntax

<SORTDESC

Notes

This command determines the order in which items are sorted in member commands in the report specification. You use this command prior to the other sort commands including SORTALTNAMES, SORTGEN, SORTLEVEL and SORTMBRNames. With the SORTDESC command, all members are sorted in descending order starting with either the letter "z" or the highest generation and moving toward the letter "a" or the lowest generation.

This command must precede the selection commands, for example CHILDREN or DESCENDANTS. If no sorting commands are used, members are output in hierarchical order based on the member outline. Any sort command remains in effect until another sort command is issued.

Example

The following example is based on Sample Basic.

```
<PAGE (Market, Measures)
Massachusetts Sales
<COLUMN (Scenario, Year)
Actual Budget
Jan Feb Mar
<ROW (Product)
<SORTDESC
<ICHILDREN Product
!
```

This example produces the following report:

Massachusetts Sales						
	Actual			Budget		
	Jan	Feb	Mar	Jan	Feb	Mar
Product	=====	=====	=====	=====	=====	=====
Diet	1,251	1,206	1,203	1,170	1,130	1,120
400	#Missing	#Missing	#Missing	#Missing	#Missing	#Missing
300	160	136	132	160	140	130
200	130	132	129	100	100	100
100	467	468	450	450	450	430
	494	470	492	460	440	460

See Also

- [ALLINSAMEDIM](#)
- [DESCENDANTS](#)
- [SORTASC](#)
- [SORTALTNAMES](#)
- [SORTGEN](#)
- [SORTLEVEL](#)
- [SORTMBRNames](#)
- [SORTNONE](#)

SORTGEN

Sorts all members added with a member command, such as <CHILDREN, according to the generation of the member in the Database Outline. The top of the dimension in the Outline is generation 1 for the dimension. The children of the top are generation 2, and so on. Each member's generation is one higher than its parent. Members entered directly in the report specification without using a member selection command, calculated rows and column names, or member commands encountered in the specification prior to the SORTGEN command, are not affected by the command.

This command must precede the selection commands, for example CHILDREN or DESCENDANTS. If no sorting commands are used, members are output in hierarchical order based on the member outline. Any sort command remains in effect until another sort command is issued.

Syntax

```
<SORTGEN
```

Notes

- SORTGEN sorts members from the last generation, which is the leaf member of the dimension, to the first generation in the branch, which is the root of the dimension.
- SORTGEN is not affected by other sort commands.

Example

The following example is based on Sample Basic.

```
<PAGE (Product, Measures)
East Sales
<COLUMN (Scenario, Year)

Actual Budget
Jan Feb Mar
<ROW (Market)
<SORTGEN
<IDESCENDANTS Market
!
```

Which produces the following report:

	Product Sales					
	Actual			Budget		
	Jan	Feb	Mar	Jan	Feb	Mar
	=====	=====	=====	=====	=====	=====
Market	31,538	32,069	32,213	29,480	30,000	30,200
East	6,780	6,920	6,921	6,180	6,350	6,360
West	10,436	10,564	10,674	9,460	9,530	9,640
South	3,976	4,082	4,055	3,870	3,970	3,990
Central	10,346	10,503	10,563	9,970	10,150	10,210
New York	2,479	2,625	2,601	2,300	2,450	2,440
Massachusetts	1,251	1,206	1,203	1,170	1,130	1,120
Florida	1,321	1,383	1,428	1,170	1,250	1,290

Connecticut	1,197	1,157	1,118	1,080	1,040	1,000
New Hampshire	532	549	571	460	480	510
California	3,602	3,699	3,755	3,450	3,490	3,570
Oregon	1,741	1,667	1,650	1,590	1,530	1,500
Washington	1,605	1,629	1,601	1,420	1,450	1,440
Utah	1,388	1,397	1,424	1,320	1,320	1,350
Nevada	2,100	2,172	2,244	1,680	1,740	1,780
Texas	1,455	1,544	1,506	1,490	1,580	1,560
Oklahoma	980	980	1,001	920	920	940
Louisiana	978	980	948	900	910	900
New Mexico	563	578	600	560	560	590
Illinois	2,538	2,653	2,697	2,580	2,690	2,740
Ohio	1,471	1,411	1,390	1,470	1,410	1,380
Wisconsin	1,341	1,363	1,369	1,280	1,330	1,330
Missouri	1,009	1,014	1,039	960	980	1,000
Iowa	2,029	2,042	2,104	1,810	1,800	1,860
Colorado	1,958	2,020	1,964	1,870	1,940	1,900

See Also

- [ALLINSAMEDIM](#)
- [CHILDREN](#)
- [DESCENDANTS](#)
- [SORTASC](#)
- [SORTALTNAMES](#)
- [SORTDESC](#)
- [SORTLEVEL](#)
- [SORTMBRNames](#)
- [SORTNONE](#)

SORTLEVEL

Sorts all members added with a member selection command, such as <CHILDREN, according to the level of the member.

Each member is 1 level higher than the highest level of its children. Members entered without using a member selection command, calculated rows and column names, or member commands encountered prior to the SORTLEVEL command are not affected.

This command must precede the selection commands, for example CHILDREN or DESCENDANTS.

Syntax

<SORTLEVEL

Notes

SORTLEVEL sorts members from the lowest level to the highest level.

Example

The following example is based on Sample Basic.

```
<PAGE (Product, Measures)
East Sales
```

<COLUMN (Scenario, Year)

Actual Budget

Jan Feb Mar

<ROW (Market)

<SORTLEVEL

<IDESCENDANTS Market

!

This example produces the following report:

Product Sales

	Actual			Budget		
	Jan	Feb	Mar	Jan	Feb	Mar
	=====	=====	=====	=====	=====	=====
New York	2,479	2,625	2,601	2,300	2,450	2,440
Massachusetts	1,251	1,206	1,203	1,170	1,130	1,120
Florida	1,321	1,383	1,428	1,170	1,250	1,290
Connecticut	1,197	1,157	1,118	1,080	1,040	1,000
New Hampshire	532	549	571	460	480	510
California	3,602	3,699	3,755	3,450	3,490	3,570
Oregon	1,741	1,667	1,650	1,590	1,530	1,500
Washington	1,605	1,629	1,601	1,420	1,450	1,440
Utah	1,388	1,397	1,424	1,320	1,320	1,350
Nevada	2,100	2,172	2,244	1,680	1,740	1,780
Texas	1,455	1,544	1,506	1,490	1,580	1,560
Oklahoma	980	980	1,001	920	920	940
Louisiana	978	980	948	900	910	900
New Mexico	563	578	600	560	560	590
Illinois	2,538	2,653	2,697	2,580	2,690	2,740
Ohio	1,471	1,411	1,390	1,470	1,410	1,380
Wisconsin	1,341	1,363	1,369	1,280	1,330	1,330
Missouri	1,009	1,014	1,039	960	980	1,000
Iowa	2,029	2,042	2,104	1,810	1,800	1,860
Colorado	1,958	2,020	1,964	1,870	1,940	1,900
East	6,780	6,920	6,921	6,180	6,350	6,360
West	10,436	10,564	10,674	9,460	9,530	9,640
South	3,976	4,082	4,055	3,870	3,970	3,990
Central	10,346	10,503	10,563	9,970	10,150	10,210
Market	31,538	32,069	32,213	29,480	30,000	30,200

See Also

- [ALLINSAMEDIM](#)
- [CHILDREN](#)
- [DESCENDANTS](#)
- [SORTASC](#)
- [SORTALTNAMES](#)
- [SORTDESC](#)
- [SORTGEN](#)
- [SORTMBRNames](#)
- [SORTNONE](#)

SORTMBRNames

Sorts all members added with a member selection command, such as <CHILDREN alphabetically by member name when the members are added to the report. Members entered without using a member selection command, calculated rows and column names, or member commands encountered in the specification prior to the SORTMBRNames command are not affected.

This command must precede the selection commands. Any sort command remains in effect until another sort command is issued.

Syntax

<SORTMBRNames

Notes

- SORTMBRNames disregards hierarchical relationships between members.
- Numeric characters rise above alphanumeric characters in the sort order. For example, 100 rises above A200, which rises above Accounts.
- If [SORTASC](#) or [SORTDESC](#) are used to control sorting, they must precede the SORTMBRNames command.

Example

The following example is based on Sample Basic.

```
<PAGE (Product, Measures)
Sales
<COLUMN (Scenario, Year)
Actual Budget
Jan Feb Mar
<ROW (Market)
<SORTMBRNames
<IDESCENDANTS South
!
```

This example produces the following report:

	Actual			Budget		
	Jan	Feb	Mar	Jan	Feb	Mar
Louisiana	978	980	948	900	910	900
New Mexico	563	578	600	560	560	590
Oklahoma	980	980	1,001	920	920	940
South	3,976	4,082	4,055	3,870	3,970	3,990
Texas	1,455	1,544	1,506	1,490	1,580	1,560

SORTNONE

Disables all previous sorting commands.

Syntax

<SORTNONE

Notes

This command disables all previous sorting commands so that members added to the report with member selection commands are added in outline order.

See Also

- [ALLINSAMEDIM](#)
- [DESCENDANTS](#)
- [SORTALT NAMES](#)
- [SORTDESC](#)
- [SORTGEN](#)
- [SORTLEVEL](#)
- [SORTMBR NAMES](#)

SPARSE

Tells Essbase to use the sparse data extraction method, which optimizes performance when a high proportion of the reported data rows are #MISSING. Essbase cannot use the sparse data retrieval optimization method on Dynamic Calc or Dynamic Calc and Store members.

If you have at least one sparse row dimension in your report, Essbase uses the sparse data extraction method in two cases:

- Case 1: You use SUPMISSINGROWS in your report script to suppress #MISSING values, and Essbase estimates that a very high proportion of the requested data rows are #MISSING. In this case, Essbase implicitly uses the sparse method to optimize performance.
- Case 2: You explicitly use the SPARSE command in your report script. This forces Essbase to use the sparse method. If you use the SPARSE command in a report, and you have not used SUPMISSINGROWS, Essbase automatically turns on SUPMISSINGROWS for the report containing SPARSE. Essbase also turns on SUPMISSINGROWS for all following reports in your report script, unless you specify INCMISSINGROWS in a subsequent report.

Note: If your report does not contain at least one sparse row dimension, Essbase cannot use the sparse method, and reverts to the regular method. Essbase displays a message to tell you that it cannot use the sparse method.

When Essbase uses the sparse method, it displays the following message: Report Writer Sparse Extractor method will be executed.

If you have at least one sparse row dimension in your report, the report is very large, and a very high proportion of the reported data rows are #MISSING, you may want to use the SPARSE command. You can then assess if this improves your report script performance.

If your report requests a small number of cells (#MISSING and non-missing), the sparse data extraction method is less efficient than the regular method. In this case, Essbase uses the regular

method, unless you have at least one sparse row dimension in your report, and you explicitly use the SPARSE command.

SPARSE method: When Essbase uses the sparse data extraction method, Essbase first selects the row member combinations you have requested in your report script. Essbase looks at only the non-missing data blocks for these row member combinations. If your database is very sparse, this method is very efficient.

Regular method: By contrast, when Essbase uses the regular data extraction method, it cycles through every possible member combination requested by the report script. It then reports only those rows that are *not*#MISSING.

For example, suppose that only 1 in 10,000 data cells exist in a database. The remaining cells are #MISSING. On this database, you run a report script that requests 100% of the data, and uses SUPMISSINGROWS to suppress the #MISSING values.

If Essbase uses the regular method of data extraction, it cycles through all the requested member combinations.

If Essbase uses the sparse extraction method, it looks only at the non-missing data blocks for the row member combinations requested. As this database is very sparse, the number of data blocks is probably low. The sparse method produces the report much faster.

To exclude the sparse data extraction method from being used, use the <SPARSEOFF command. For example, you might want to use this command when reporting on data that includes Dynamic Calc and Dynamic Calc and Store members.

Syntax

<SPARSE

<SPARSEOFF

Notes

- The sparse extraction method cannot be used if the report contains attribute dimensions.
- When you include multiple logical reports separated by a ! within one report script, include the format commands/Headings for each logical report.

See Also

- [SUPMISSINGROWS](#)

STARTHEADING

Starts the definition of the page heading in place of the default heading, which is displayed at the top of each page in the report or immediately following a HEADING command.

Syntax

{ STARTHEADING }

Notes

This command starts the definition of the page heading in place of the default heading, which is displayed at the top of each page in the report or immediately following a HEADING command. The ENDHEADING command signifies the end of the heading; all commands encountered between the STARTHEADING and ENDHEADING are part of the heading definition. Unless SUPHEADING is used outside the STARTHEADING / ENDHEADING group, the commands within the STARTHEADING/ENDHEADING group are re-executed at the start of each new page.

By default, new pages are started whenever a page member changes, the makeup of column headings change, the page length is exceeded and SUPFEED has not been used, the NEWPAGE command is issued, the HEADING command is issued, or the PAGEONDIMENSION command causes a page break. A custom heading will include the default page header and column headers unless they are suppressed with SUPPAGEHEADING and/or SUPCOLHEADING in the custom heading definition.

Note that headings (whether the default page and column headings or a custom heading created with ENDHEADING) do not get output right at the start of a new page. They are delayed until the next non-suppressed output data row is encountered, and even then the heading is output only after the data row's format { } commands have been processed. This avoids blank pages with nothing but headers on them but it can make it awkward to put out a TEXT (or other format which produces output) between the heading and the first output data row.

Tip: To ensure that headings display correctly, structure the report script so that column member selections precede row member selections, and make sure that the script contains at least one column member.

Default Value

Replaces default heading.

Example

The following example shows how to define a heading for a report. All the commands within the STARTHEADING and ENDHEADING commands are executed at the top of each page. The TEXT commands display information about the person who prepared the report, the date the report was generated, and other title information.

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual

    <COLUMN (Year)
    <CHILDREN Year

<ROW (Product)

{ STARTHEADING TEXT 2 "Prepared by:" 14 "*USERNAME"
  C "The Electronics Club" 60 "*PAGESTRING"
  TEXT C "Quarterly Sales by City" 60 "*DATE"
  SUPPAGEHEADING
  TEXT 2 "*PAGEHDR" SKIP ENDHEADING}
```

<IDESCENDANTS Product
!

This example produces the following report:

Prepared by: Bob The Electronics Club Page: 1
 Quarterly Sales by City 05/13/03

Chicago Sales Actual

	Qtr1	Qtr2	Qtr3	Qtr4
	=====	=====	=====	=====
Stereo	2,591	2,476	2,567	3,035
Compact_Disc	3,150	3,021	3,032	3,974
Audio	5,741	5,497	5,599	7,009
Television	4,410	4,001	4,934	6,261
VCR	3,879	3,579	4,276	4,877
Camera	2,506	2,522	2,602	3,227
Visual	10,795	10,102	11,812	14,365
Product	16,536	15,599	17,411	21,374

See Also

- [ENDHEADING](#)
- [HEADING](#)
- [IMMHEADING](#)
- [SUPCOLHEADING](#)
- [SUPHEADING](#)
- [SUPPAGEHEADING](#)

SUDA

Selects members based on a common attribute, defined as a UDA (user-defined attribute) along with their shared counterparts.

Syntax

<SUDA (*dimName*, *udaStr*)

Parameter Description

dimName Name of the dimension associated with *udaStr*.

udaStr Name of the UDA.

Notes

- You can use the <SUDA command as a standalone command or as a selection command inside the [LINK](#) statement.
- You cannot use attributes as arguments.

- With the <UDA command, Report Extractor selects only the members tagged with the specified UDA. Shared members are not selected. For example, consider the following outline structure:

```
Product
  100
    100-10
    100-20 (UDAS: No Carb)
  200
    200-10
    200-20 (UDAS: No Carb)
Diet
  100-20 (shared)
  200-20 (shared)
```

The following command returns no members because the children of Diet are not recognized as having the UDA "No Carb":

```
<CHILDREN (Diet) and <UDA (Product, "No Carb")
```

In contrast, the <SUDA report command enables Report Extractor to recognize all instances of shared members as having the UDA associated with the original instance of the member. For example, the following command:

```
<CHILDREN (Diet) and <SUDA (Product, "No Carb")
```

returns the following members:

```
[Product].[100].[100-20]
[Product].[200].[200-20]
[Product].[Diet].[100-20]
[Product].[Diet].[200-20]
```

because these members are children of Diet, and the "No Carb" UDA associated with the first instances of the members is also associated with the shared members.

Example

The following example uses the SUDA command within a LINK statement to select shared members under Diet that are not "No Carb":

```
<LINK (<DESC(Diet) and not <SUDA (product, "No Carb))
```

See Also

- [UDA](#)

SUPALL

Suppresses the display of the page and column headings, all member names, page breaks, commas, and brackets.

Syntax

```
{ SUPALL }
```

Notes

With this command, you see the data of the report and any text displayed as the result of the TEXT command. This command is equivalent to SUPHEADING, SUPPAGEHEADING, SUPCOLHEADING, SUPNAMES, SUPBRACKETS, SUPFEED, and SUPCOMMAS.

Example

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual
  <COLUMN (Year)
  <CHILDREN Year
<ROW (Product)
<ICHILDREN Audio
  !

{ SUPALL }
Boston Sales Actual
  <CHILDREN Year
<ICHILDREN Audio
  !
```

This example produces the following report.

Note: The last three rows show the totals for Boston, without headings.

```
Chicago Sales Actual

      Qtr1   Qtr2   Qtr3   Qtr4
      =====
Stereo      2,591  2,476  2,567  3,035
Compact_Disc 3,150  3,021  3,032  3,974
  Audio      5,741  5,497  5,599  7,009

      2450    2341    2377    2917
      3290    3034    3132    3571
      5740    5375    5509    6488
```

See Also

- [SUPBRACKETS](#)
- [SUPCOLHEADING](#)
- [SUPCOMMAS](#)
- [SUPCURHEADING](#)
- [SUPEMPTYROWS](#)
- [SUPEUROPEAN](#)
- [SUPFEED](#)
- [SUPHEADING](#)
- [SUPMISSINGROWS](#)
- [SUPNAMES](#)
- [SUPPAGEHEADING](#)
- [SUPZEROROWS](#)

SUPBRACKETS

Suppresses the display of parentheses around negative numbers.

Syntax

```
{ SUPBRACKETS }
```

Notes

The negative sign,(-), rather than parentheses, indicates negative numbers.

Example

```
{ SUPBRACKETS }
```

displays (34.43) as -34.43.

See Also

- [COMMAS](#)
- [DECIMAL](#)
- [SUPALL](#)
- [SUPBRACKETS](#)
- [SUPCOMMAS](#)

SUPCOLHEADING

Suppresses display of default column headings.

Syntax

```
{ SUPCOLHEADING }
```

Notes

Unless a custom heading is defined, you will see only the page heading members at the top of the page and row members on the left side of each row. The keyword >*COLHDR with the TEXT command is not affected by SUPCOLHEADING and may still be used to generate column headings where desired.

Example

```
<PAGE (Market, Accounts, Scenario)
{ SUPCOLHEADING }
Boston Sales Actual
    <COLUMN (Year)
    <CHILDREN Year
<ROW (Product)
<ICHILDREN Audio
!
```

This example produces the following report:

```
Boston Sales Actual
```

Stereo	2,450	2,341	2,377	2,917
Compact_Disc	3,290	3,034	3,132	3,571
Audio	5,740	5,375	5,509	6,488

See Also

- [COLHEADING](#)
- [NAMESON](#)
- [PAGEHEADING](#)
- [SUPNAMES](#)
- [SUPPAGEHEADING](#)

SUPCOMMAS

Suppresses the display of commas in numbers greater than 999.

Note: The display of commas is the default.

Syntax

```
{ SUPCOMMAS }
```

Example

```
{ SUPCOMMAS }
```

displays the number 12,234,534.23 as 12234534.23.

See Also

- [BRACKETS](#)
- [COMMAS](#)
- [DECIMAL](#)
- [SUPBRACKETS](#)

SUPCURHEADING

Suppresses the display of currency information when you use the CURRENCY command to convert the data values in your report to a specified currency.

Syntax

```
{ SUPCURHEADING }
```

Notes

The keyword *CURRENCY with the TEXT command is not affected by SUPCURHEADING and may be used after SUPCURHEADING to create custom currency heading and placement.

See Also

- [CURHEADING](#)
- [CURRENCY](#)

SUPEMPTYROWS

Suppresses the display of rows that have only 0 or #MISSING values in the row.

Syntax

```
{ SUPEMPTYROWS }
```

Notes

This command suppresses the display of zero rows, for example, rows that have only 0 or missing values. The report will contain only rows which have at least one data value which is neither #MISSING nor zero.

Example

{SUPEMPTYROWS} would suppress the display of the following row in a report:

```
Qtr1 Actual 0 #Missing 0 0 #Missing
```

See Also

- [INCEMPTYROWS](#)
- [INCMISSINGROWS](#)
- [INCZEROROWS](#)
- [SUPMISSINGROWS](#)
- [SUPZEROROWS](#)

SUPEUROPEAN

Disables the European method for displaying numbers.

Syntax

```
{ SUPEUROPEAN }
```

Notes

In European mode, commas separate the decimal and whole number portion of a data value, while decimal points are used for the thousands separator character. Non-European number display uses commas to separate thousands and the decimal point to separate decimals.

SUPEUROPEAN need only be used after a EUROPEAN command.

Default Value

Non-European is the default.

Example

See the example for EUROPEAN.

See Also

- [EUROPEAN](#)

SUPFEED

Suppresses the automatic insertion of a physical page break whenever the number of lines on a page exceeds the current PAGELENGTH setting.

Syntax

```
{ SUPFEED }
```

Notes

This command disables the FEEDON command. The command FEEDON re-enables physical page breaks. The default page length is 66 lines unless reset with the PAGELENGTH command.

Default Value

Default when performing ad-hoc reports into a spreadsheet.

See Also

- [FEEDON](#)
- [NEWPAGE](#)
- [PAGELENGTH](#)

SUPFORMATS

Suppresses formats that produce extra output such as underlines and skips.

Syntax

```
{ SUPFORMATS }
```

Notes

The SUPFORMATS command is used in those instances where you need to suppress formats which produce output, such as underlines, skips, etc., because the data row with which the formats are associated is automatically (and therefore unpredictably) suppressed due to commands such as SUPMISSING. Otherwise, a page could be filled with "orphan" underlines and no data. If you want to retain formatting in this case, you need to turn the formats on by using the INCFORMATS command.

Default Value

Set to "ON" by default when the SUPMASK, SUPMISSING, or SUPZERO commands are used.

See Also

- [INCFORMATS](#)

SUPHEADING

Suppresses the display of the default heading (page header and column headers) or custom header, if defined, at the top of each page.

Syntax

```
{ SUPHEADING }
```

Notes

A custom heading is defined with the STARTHEADING and ENDHEADING commands. The HEADING command cancels the effect of the SUPHEADING command in addition to displaying the heading immediately prior to the next non-suppressed data row to be output. By default, new pages are started either when a page member changes, the makeup of column headings change, the page length is exceeded and SUPFEED has not been used, the NEWPAGE command is issued, the HEADING command is issued, or the PAGEONDIMENSION command causes a page break.

Default Value

Display of the default heading is suppressed.

Example

See the example for STARTHEADING.

See Also

- [ENDHEADING](#)
- [HEADING](#)
- [IMMHEADING](#)
- [STARTHEADING](#)

SUPMASK

Suppresses the display of a text mask.

Syntax

```
{ SUPMASK }
```

Notes

Text masks are defined using the MASK command. The MASK command cancels the effect of the SUPMASK command, in addition to defining a new mask. While SUPMASK is in effect, a mask text string may still be output using the TEXT command's *MASK option.

See Also

- [MASK](#)
- [TEXT](#)

SUPMISSINGROWS

Suppresses the display of rows that contain only #MISSING values.

Syntax

```
{ SUPMISSINGROWS }
```

Example

```
<Sym  
  <Column (Scenario, Year)  
    Actual Budget  
    Jan Dec  
<Top ("Measures", 5, @DataCol(4))  
<Row (Measures, Market, Product)  
{SupMissingRows}  
  
<Idescendants Profit  
<Ichildren Market  
<Idescendants Product  
!
```

This example produces the following report:

			Actual		Budget	
			Jan	Dec	Jan	Dec
			=====	=====	=====	=====
Sales	Market	Product	31,538	33,342	29,480	30,820
Margin	Market	Product	17,378	18,435	16,850	17,360
COGS	Market	Product	14,160	14,907	12,630	13,460
Sales	Central	Product	10,346	10,662	9,970	10,310
	West	Product	10,436	11,116	9,460	10,200

See Also

- [INCEMPTYROWS](#)
- [INCMISSINGROWS](#)
- [INCZEROROWS](#)
- [SUPEMPTYROWS](#)
- [SUPZEROROWS](#)

SUPNAMES

Suppresses the display of row member names in the final report.

Syntax

```
{ SUPNAMES }
```

Notes

The NAMESON command re-enables the display of row member names in the report.

Example

The following example is based on Demo Basic.

```
<PAGE (Market, Accounts, Scenario)  
Chicago Sales Actual  
  
  <COLUMN (Year)
```

```

    <CHILDREN Year

<ROW (Product)
<ICHILDREN Audio
    !

{ SUPNAMES }
Boston Sales Actual
    <CHILDREN Year
<ICHILDREN Audio
    !

```

This example produces the following report:

Note: The rows with the suppressed row member names are not indented with whitespace.

```

                Chicago Sales Actual
                Qtr1    Qtr2    Qtr3    Qtr4
                =====
Stereo          2,591    2,476    2,567    3,035
Compact_Disc   3,150    3,021    3,032    3,974
  Audio        5,741    5,497    5,599    7,009
                Boston Sales Actual
                Qtr1    Qtr2    Qtr3    Qtr4
                =====
2,450    2,341    2,377    2,917
3,290    3,034    3,132    3,571
5,740    5,375    5,509    6,488

```

See Also

- [COLHEADING](#)
- [NAMESON](#)
- [PAGEHEADING](#)
- [SUPCOLHEADING](#)
- [SUPPAGEHEADING](#)

SUOUTPUT

Suppresses all output, except columns, while continuing to process other operations such as calculations or format settings. Use the OUTPUT command to resume output.

Syntax

```
{ SUOUTPUT }
```

Example

```

<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual

    <COLUMN (Year)
    <CHILDREN Year

```

```

<ROW (Product)
<ICHILDREN Audio
Stereo
Compact_Disc
{SUOUTPUT}
VCR
TELEVISION
{OUTPUT}
Audio
!
{ SUPNAMES }
Boston Sales Actual
    <CHILDREN Year
<ICHILDREN Audio
!

```

Which produces the same report as in the [SUPNAMES](#) example.

See Also

- [OUTPUT](#)

SUPPAGEHEADING

Suppresses display of the page member heading whenever a heading is generated.

Syntax

```
{ SUPPAGEHEADING }
```

Notes

This command does not suppress column headings and row members.

To reinstate page headings, use the PAGEHEADING command.

The keyword *PAGEHDR with the TEXT command may be used after a SUPPAGEHEADING to produce a custom page member heading. *PAGEHDR with the TEXT is not affected by SUPCOLHEADING.

Example

```

<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual

    <COLUMN (Year)
    <CHILDREN Year

<ROW (Product)
<ICHILDREN Audio
!
{ SUPPAGEHEADING }
Boston Sales Actual
    <CHILDREN Year
<ICHILDREN Audio

```

!

This example produces the following report:

```
                Chicago Sales Actual

                Qtr1   Qtr2   Qtr3   Qtr4
                =====
Stereo          2,591   2,476   2,567   3,035
Compact_Disc    3,150   3,021   3,032   3,974
  Audio         5,741   5,497   5,599   7,009

                Qtr1   Qtr2   Qtr3   Qtr4
                =====
Stereo          2,450   2,341   2,377   2,917
Compact_Disc    3,290   3,034   3,132   3,571
  Audio         5,740   5,375   5,509   6,488
```

See Also

- [COLHEADING](#)
- [HEADING](#)
- [IMMHEADING](#)
- [NAMESON](#)
- [PAGEHEADING](#)
- [SUPCOLHEADING](#)
- [SUPNAMES](#)
- [TEXT](#)

SUPSHARE

Suppresses the display of later instances of shared members when you use generation or level names to extract data for your report.

Syntax

```
<SUPSHARE
```

Notes

This command suppresses the display of later instances of shared members only when you extract data using:

- Default or user-defined generation or level names
- DIMBOTTOM
- OFSAMEGEN
- ONSAMELEVELAS

SUPSHARE suppresses the display for the duration of the script, which can contain one or more reports. Use the SUPSHAREOFF command to reinstate the display of shared members.

Default Value

SUPSHAREOFF.

Example

The Sample Basic database has a shared level of diet drinks. The shared members are 100-20 (Diet Cola), 200-20 (Diet Root Beer), and 300-30 (Diet Cream). All are level 0 members on the Product dimension. The following report:

```
{SUPMISSINGROWS}
<SUPSHARE
<PAGE (Measures, Market, Scenario)
Sales West Actual
<COLUMN (Year)
<IDESCENDANTS Qtr1
<ROW (Product)
lev0, Product
!
```

returns the following data. The shared members appear only once in the data.

	Sales West Actual			
	Jan	Feb	Mar	Qtr1
	=====	=====	=====	=====
100-10	1,174	1,146	1,173	3,493
100-20	700	726	727	2,153
100-30	465	426	413	1,304
200-10	667	705	707	2,079
200-20	1,203	1,209	1,209	3,621
200-30	853	845	880	2,578
300-10	1,102	1,127	1,133	3,362
300-20	523	546	566	1,635
300-30	977	1,029	1,040	3,046
400-10	1,115	1,122	1,107	3,344
400-20	1,032	1,065	1,100	3,197
400-30	625	618	619	1,862

See Also

- [SUPSHAREOFF](#)

SUPSHAREOFF

The SUPSHAREOFF command reinstates the display of later instances of shared members after they have been suppressed using the SUPSHARE command.

Syntax

```
<SUPSHAREOFF
```

Notes

You can suppress and reinstate shared member display only when you extract data for your report using:

- Default or user-defined generation or level names

- DIMBOTTOM
- OFSAMEGEN
- ONSAMELEVELAS

Default Value

SUPSHAREOFF.

Example

The Sample Basic database has a shared level of diet drinks. The shared members are 100-20 (Diet Cola), 200-20 (Diet Root Beer), and 300-30 (Diet Cream). All are level 0 members on the Product dimension. The following report:

```
{SUPMISSINGROWS}
<SUPSHAREOFF
<PAGE (Measures, Market, Scenario)
Sales West Actual
<COLUMN (Year)
<IDESCENDANTS Qtr1
<ROW (Product)
lev0, Product
!
```

returns the following data. The example assumes that you have used SUPSHARE in a previous report in the report script. The SUPSHAREOFF command reinstates the shared member display so that the shared members appear twice in the report.

	Sales West Actual			
	Jan	Feb	Mar	Qtr1
	=====	=====	=====	=====
100-10	1,174	1,146	1,173	3,493
100-20	700	726	727	2,153
100-30	465	426	413	1,304
200-10	667	705	707	2,079
200-20	1,203	1,209	1,209	3,621
200-30	853	845	880	2,578
300-10	1,102	1,127	1,133	3,362
300-20	523	546	566	1,635
300-30	977	1,029	1,040	3,046
400-10	1,115	1,122	1,107	3,344
400-20	1,032	1,065	1,100	3,197
400-30	625	618	619	1,862
100-20	700	726	727	2,153
200-20	1,203	1,209	1,209	3,621
300-30	977	1,029	1,040	3,046

See Also

- [SUPSHARE](#)

SUPZEROROWS

The SUPZEROROWS command suppresses the display of rows that have only 0 values.

Syntax

```
{ SUPZEROROWS }
```

Example

{SUPZEROROWS} would not display the following row in the report:

```
Qtr1 Actual 0 0 0 0
```

but would display the following row:

```
Qtr1 Actual 0 #Missing 0 0
```

See Also

- [INCEMPTYROWS](#)
- [INCZEROROWS](#)
- [SUPEMPTYROWS](#)
- [SUPMISSINGROWS](#)

SYM

Forces a symmetric report, regardless of the data selection. Use SYM to change the symmetry of a report that Essbase would create as an asymmetric report.

Syntax

```
<SYM
```

Notes

This command is used to set the report type as symmetric. Under default conditions (for example, when neither the ASYM nor SYM commands have been used), Essbase will print an asymmetric report (with BLOCKHEADERS) when all column dimensions include the same number of selected members and all members for each column dimension are on the same line. Otherwise, a symmetric report (with PYRAMIDHEADERS) is produced. If the <SYM keyword is used, all report headers will appear in a symmetric format, even if there are equal numbers of members in each row of the column header. A symmetric report will also result if at least one of the column member lists is broken out onto more than one line.

When the <SYM keyword is used, the report will always be generated as a symmetric report, even with equal numbers of members selected in each column dimension. This is especially useful when you want to create a symmetric report without having to repeatedly type the lower-level members of symmetric/asymmetric reports. For a more detailed explanation see the <ASYM command. To turn off symmetric-only mode, use the <ASYM command.

Default Value

Essbase prints a symmetric report (with PYRAMIDHEADERS) when column dimensions do not include the same number of selected members or the members for each column dimension are not on the same line.

Example

The following example is based on Sample Basic.

```
<PAGE (Measures, Market)
Texas Sales
<SYM
    <COLUMN (Scenario, Year)
        Actual Budget
        Jan Feb
<ROW (Product)
<IDESCENDANTS "100"
!
```

This example produces the following report:

```
                Sales Texas
                Actual      Budget
                Jan      Feb      Jan      Feb
                =====
100-10          452      465      560      580
100-20          190      190      230      230
100-30          #Missing #Missing #Missing #Missing
    100          642      655      790      810
```

See Also

- [ASYM](#)

TABDELIMIT

The TABDELIMIT command places tabs rather than spaces between columns.

Syntax

```
{ TABDELIMIT }
```

Notes

This command is useful when you want to turn report output into a more compressed form for export. TABDELIMIT can occur anywhere in a report script.

Example

```
<PAGE (Scenario)
<COLUMN (Year)
<ROW (Product, Market, Measures)
{Tabdelimit}
{ROWREPEAT}
<CHILDREN Year
<DIMBOTTOM Product
<DIMBOTTOM Market
<CHILD Profit
!
```

This example produces the following report (example truncated):

```

Scenario
Qtr1 Qtr2 Qtr3 Qtr4 Year

100-10 New York Margin 1,199 1,416 1,568 1,184 5,367
100-10 New York Total Expenses 433 488 518 430 1,869
100-10 Massachusetts Margin 1,237 1,533 1,741 1,224 5,735
100-10 Massachusetts Total Expenses 164 155 149 162 630
100-10 Florida Margin 372 442 494 375 1,683
100-10 Florida Total Expenses 174 192 200 175 741
100-10 Connecticut Margin 567 481 425 557 2,030
100-10 Connecticut Total Expenses 217 197 184 215 813
100-10 New Hampshire Margin 213 249 276 209 947
100-10 New Hampshire Total Expenses 139 149 155 137 580
100-10 California Margin 1,199 1,416 1,568 1,184 5,367
100-10 California Total Expenses 433 488 517 431 1,869
100-10 Oregon Margin 270 203 202 216 891
100-10 Oregon Total Expenses 193 183 176 180 732

```

The following is the same report without TABDELIMIT:

```

<PAGE (Scenario)
<COLUMN (Year)
<ROW (Product, Market, Measures)
{ROWREPEAT}
<CHILDREN Year
<DIMBOTTOM Product
<DIMBOTTOM Market
<CHILD Profit
!
```

Without TABDELIMIT, the report looks like this (example truncated):

```

                                Scenario
                                Qtr1   Qtr2   Qtr3   Qtr4   Year
                                =====
100-10 New York Margin          1,199   1,416   1,568   1,184   5,367
100-10 New York Total Expenses  433     488     518     430     1,869

```

TEXT

Inserts text or other information on a new line in the report. You specify the character position (*charPosition*) to begin the text along with the text (*text*) that you want to display. The command can accept multiple sets of *charPosition* and *text* arguments.

In addition to text, you can use this command to insert special information based on keywords into the report. These keywords begin with "*" and must be entered exactly. For example, you can display the current date and time, the page number, or information such as user name and application.

The following list presents the keywords and associated display information.

- APPNAME: Name of application
- ARBOR: Version information

- **CALC:** All or part of a calculated row. Optionally, the CALC keyword can include an integer to designate a data column that is to be displayed. For example, `{TEXT 25 "*CALC 2" "TotSales" }` would display the column 2 value of the calculated row "TotSales" starting at character position 25, using the current column format settings in effect for column 2.

Note: Names columns are not allowed.

- **COLHDR *number1 number2*:** Displays the column heading members from the current default heading. You can indicate which rows of the column header members you want to display and which members in the row following the keyword.

Number1 selects the row of column members and *number2* selects the member within the row. If you specify just *COLHDR or *COLHDR with *number1*, the column heading members can not be combined with any other text on the same line. Furthermore, the position of the text is ignored (the header line will automatically be lined up with the existing data column setup), unless you specify both *number1* and *number2*. For example, *COLHDR 2 would display the second row of column heading members in normal position over the data columns. *COLHDR 2 5 would display the 5th column member from the second row of column heading members. This command is usually used with SUPHEADING or SUPCOLHEADING.

Using both *number1* and *number2*,

```
TEXT 25 "*COLHDR 2 3"
```

would display the third member of the column heading range from the second row of column members starting in position 25.

Generally all column heading rows after the first level in symmetric reports have repeating groups of the same range of members.

The *number2* specified refers to the member in the basic group of repeating members. For example, if Qtr1 Qtr2 and Qtr3 are the basic group which repeats in the second level column heading, the value for *number2* can range from 1 to 3. Just because the group repeats 2 or 3 times does not mean that *number2* can range up to 6 or 9. In this example, any *number2* higher than 3 would be interpreted as trying to access a calculated column header.

Calculated column headers may also be accessed by the *COLHDR option. If a report has, for example, 3 calculated columns, the *number2* which is used to access any particular level of the calculated column name depends on the number of members in the primary column header group for that heading level. In the previous example, where the second column heading line contained three members (Qtr1, Qtr2, and Qtr3), the second-level calculated column headings would be accessed with *number2* set to 4, 5, or 6 (assuming only one row names column). Again, it does not matter how many times Qtr1, Qtr2, and Qtr3 may have been repeated on the column heading line-there are still only three members of the primary column header group.

For example, if the first calculated column defined is "YTD~PCT~TOTAL", then the second level header "PCT" could be printed with `TEXT 10 "*COLHDR 2 4"`

assuming once again that the primary column heading group on level 2 had three members and only one row name dimension. Refer to ORDER for more information about column numbering.

The ORDER command does not affect the parameters for selecting the headers. The *Number2* value is based on the original column order without regard to any reordering or truncation of columns with ORDER or FIXCOLUMNS.

- COLHDRFULL, which is the full column heading along with underlines of the column headings and a 1 line skip. The position is ignored with this keyword (the headers and underlines will be aligned automatically over the data columns as currently set up) and it can not be combined with any other text on the same line.
- CURRENCY, which is the currency conversion label which indicates which currency the data values have been converted to at report time with the CURRENCY command. Usually used with SUPCURHEADING.
- DATA, which is used to display data rows. If the command does not include a column designator, it will display all data starting at the character position. If a column number is included, only that column will be displayed. See *CALC above.
- DATE, which is the date the report was generated.
- DATETIME, which is the date followed by the time the report was generated.
- DBNAME, which is the name of the data base within the application.
- EDATE, which is the date in European (dd/mm/yy) format.
- EDATETIME, which is the date in European (dd/mm/yy) format followed by the time. Time is in 24-hour format, as hour:minute:second; for example, 14:35:02.
- MACHINE, which is the network name for the machine that is running the Essbase Server.
- PAGEHDR *number*;: Displays the default page member heading. *Number* indicates which specific page members you wish to display following the keyword. The page member text can only be combined with other text on the same line if *number* is specified. For example, TEXT C *PAGEHDR 2 would display only the second page member from the page heading members from the current default page heading. It is usually used with SUPHEADING or SUPPAGEHEADING.
- PAGENO: Page number for the current page.
- PAGESTRING: Page number preceded by the text "Page:".
- TIME: Time the report was generated.
- TIMEDATE: Time followed by the date the report was generated.
- TIMEEDATE: Time followed by the European format (dd/mm/yy) date.
- USERNAME: Name of the user generating the report.

Syntax

```
{TEXT charPosition "text " [ charPosition "text" ... ]}
```

Parameter Description

<i>charPosition</i>	Character position on the line to start the text specified in the next <i>text</i> argument. When multiple sets of <i>charPositions</i> are specified, successive <i>charPositions</i> need not be in ascending order. If the positions of two text strings cause an overlap, the last overwrites the first. "Last" is determined by left-right order in the TEXT statement, not by <i>charPosition</i> .
---------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Parameter	Description
-----------	-------------

text	Text to add to the report. Commas, tabs and multiple spaces are ignored. Maximum length: 500 characters.
------	----------------------------------------------------------------------------------------------------------

Notes

- TEXT is an output command.
- *n* must be an integer greater than or equal to zero or the letter *c* for centered. (If you specify *n* as zero, the line starts at the left margin.) You must specify a value for *n*.
- TEXT does not wrap the text specified in "text".
- You can use the * (asterisk) character to add report keywords, such as *CALC and *TIME. If * precedes an invalid keyword, Essbase displays the text that follows.

Example

- Adding the text "Golden State Bottling Division" 27 spaces from the left margin of the report. This example is based on Demo Basic.

```
{TEXT 27 "Golden State Bottling Division" }
```

- The following report lists several Examples of the TEXT command.

The first set of TEXT commands is defined in the custom heading of the report which is displayed at the top of every page.

- The command { TEXT 2 "*DATETIME" C "Annual Report" 65 "*PAGESTRING" SKIP } displays the date and time starting at character position 2 of the first line of the heading, centers the text "Annual Report" in the middle of the line, and displays the text "Page" followed by the actual page number starting at character position 65 of the first line.
- The second line of the heading is defined by the command { TEXT 2 "City: " 12 "*PAGEHDR 1" } which displays the text "City:" starting a character position 2 and then displays the first page member for the page in the report. As per the first member in the PAGE command, these members are always from the Market dimension.
- The command { TEXT 2 "Account: " 12 "*PAGEHDR 2" SKIP } for the third line of heading displays the text "City" at character position 2 followed by the page heading member from the Accounts dimension.

The TEXT commands at the end of the report display summary information about the report.

- The command { TEXT 2 "Prepared by: " 18 "*USERNAME" } displays the text "Prepared by:" at character position 2 followed by the name of the user who generated the report at character position 18.
- For the next line, the command { TEXT 2 "Server Version: " 18 "*ARBOR" } displays the text "Server Version:" at character position 2 followed by the version information.
- The third line uses the command { TEXT 2 "Application: " 18 "*APPNAME" } to display the text "Application:" at character position 2 followed by the application name.
- The final line uses the command { TEXT 2 "Database: " 18 "*DBNAME" } to display the text "Database:" at character position 2 followed by the database name.

```
{ STARTHEADING
  SUPPAGEHEADING
  TEXT 2 "*DATETIME" C "Annual Report" 65 "*PAGESTRING" SKIP
  TEXT 2 "City: " 12 "*PAGEHDR 1"
  TEXT 2 "Account: " 12 "*PAGEHDR 2" SKIP
  ENDHEADING }
```

```
<PAGE (Market, Accounts)
Chicago Sales
```

```
<COLUMN (Scenario, Year)
```

```
Actual
<CHILDREN Year
```

```
<ROW Audio
```

```
{ SKIP 2 "Prepared by: " 18 "*USERNAME" }
{ TEXT 2 "Server Version: " 18 "*ARBOR" }
{ TEXT 2 "Application: " 18 "*APPNAME" }
{ TEXT 2 "Database: " 18 "*DBNAME" }
!
```

```
09/15/03 14:14:59
```

```
Annual Report
```

```
Page: 1
```

```
City: Chicago
Account: Sales
```

	Qtr1	Qtr2	Qtr3	Qtr4
	=====	=====	=====	=====
Stereo	2,591	2,476	2,567	3,035
Compact_Disc	3,150	3,021	3,032	3,974
Audio	5,741	5,497	5,599	7,009

```
Prepared by : Admin
Server Version: Gemini Alpha - 9/6/95 [Fri Sep 15 14:14:59 1995]
Application: Demo
Database: Basic
```

- The remaining examples of the TEXT command are based on the following report heading:

```
Chicago Sales
```

	Actual			Budget		
	Qtr1	Qtr2	Qtr3	Qtr1	Qtr2	Qtr3
	=====	=====	=====	=====	=====	=====

- { TEXT 10 "*COLHDR 2" }

would produce the following line:

```
Qtr1 Qtr2 Qtr3 Qtr1 Qtr2 Qtr3
```

- { TEXT 10 "*COLHDR 2 3" }

would produce the following text at position 10:

```
Qtr3
```


- { TEXT 10 "*COLHDR 1 2" }

would produce the following text at position 10:

Budget

- { TEXT 10 "COLHDRFULL" }

would produce the following lines of text regardless of the value of *charposition*:

Actual			Budget		
Qtr1	Qtr2	Qtr3	Qtr1	Qtr2	Qtr3
=====	=====	=====	=====	=====	=====

See Also

- [SUPCOLHEADING](#)
- [SUPPAGEHEADING](#)

TODATE

The TODATE command converts date strings to numbers that can be used to extract data output for a specific time period. TODATE converts date strings into the number of seconds elapsed since midnight, January 1, 1970.

Syntax

```
<TODATE (formatString, dateString)
```

Parameter	Description
-----------	-------------

formatString	The date string format, either "mm-dd-yyyy" or "dd-mm-yyyy".
--------------	--------------------------------------------------------------

dateString	The date string.
------------	------------------

Notes

- If you specify a date that is earlier than 01-01-1970, this command returns an error.
- The latest date supported by this command is 12-31-2037.

Example

```
<TODATE ("dd-mm-yyyy", "15-10-2002")
```

See Also

- [ATTRIBUTE](#)
- [WITHATTR](#)

TOP

Returns rows with the highest values of a specified data column.

Syntax

<TOP ([<rowgroupDimension>],) <rows>, <column>)

Parameter	Description
<rowgroupDimension>	Optional. Row grouping dimension that determines the rows to sort as a set. The default is the inner row.
<rows>	Positive integer that specifies the number of rows to be returned; must be greater than 0.
<column>	@DATACOL (<colnumber>) @DATACOL (<colnumber>) where <colnumber> is the target column number; must be between 1 and the maximum number of columns in the report.

Notes

This command sorts the result set by the value of the specified data column in descending order.

Rows containing #MISSING values in the sort column are discarded from the result set before TOP is applied. You can use TOP and BOTTOM, ORDERBY and RESTRICT in the same report script, but you can use each command only once per report. If you repeat the same command in the same report script, the second command overwrites the first. Place global script formatting commands before a PAGE, COLUMN command or associated member (for example, <CHILDREN or <IDESCENDANTS). Avoid using row formatting commands with TOP.

If any of the ORDERBY, TOP, BOTTOM, or RESTRICT commands coexist in a report script, the row group dimension <rowgroupDimension> should be the same. This prevents confusion about the sorting and ordering of rows within a row group. Otherwise, an error is issued. The ORDERBY, TOP, and BOTTOM commands sort a report output by its data values. The RESTRICT command restricts the number of valid rows for the report output. Their execution order is:

1. Any sorting command that sorts on member names (for example <SORTDESC or <SORTASC)
2. RESTRICT
3. TOP and BOTTOM
4. ORDERBY

This order applies regardless of the order in which the commands appear in the report script. For an example that uses TOP, BOTTOM, ORDERBY, and RESTRICT together, see the entry for the BOTTOM command.

You can configure the size of the internal buffers used for storing and sorting the extracted data. The following settings affect the way the RESTRICT, TOP, and BOTTOM commands work:

- Retrieval Buffer Size (a database setting)
- Retrieval Sort Buffer Size (a database setting)
- “NUMERICPRECISION” on page 477 (an essbase.cfg setting)

For more information on the database settings, see the *Oracle Essbase Database Administrator's Guide*.

Example

```
<Sym
<Column (Scenario, Year)
Actual Budget
Jan Dec
<Top ("Measures", 5, @DataCol(4))
<Row (Measures, Market, Product)
{SupMissingRows}

<Idescendants Profit
<Ichildren Market
<Idescendants Product
!
```

Which produces the following report based on the Sample Basic sample database:

			Actual		Budget	
			Jan	Dec	Jan	Dec
			=====	=====	=====	=====
Sales	Market	Product	31,538	33,342	31,538	30,820
Margin	Market	Product	17,378	18,435	17,378	17,360
COGS	Market	Product	14,160	14,907	14,160	13,460
Sales	Central	Product	10,346	10,662	10,346	10,310
	West	Product	10,436	11,116	10,436	10,200

See Also

- [RESTRICT](#)
- [ORDERBY](#)
- [BOTTOM](#)

UCHARACTERS

Underlines all non-blank characters in the preceding row.

To underline names cleanly, the UCHARACTERS command treats a single space between two non-space characters as a character to underline. For example, in the name Sales_Revenue, the underscore is changed to a space on output, UCHARACTERS changes the space to "_". Default underline character "=" is used.

Syntax

```
{ UCHARACTERS [ "char" ] }
```

Parameter Description

"char" Optional. A single-byte character, enclosed in quotation marks, used as the underline character.

Notes

- Double-byte characters are not supported.

Example

The following example is based on Demo Basic.

{ UCHARACTERS } underlines all the characters in the previous (Television) row.

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual
```

```
    <COLUMN (Year)
    <ICHILDREN Year
```

```
<ROW (Product)
Television
```

```
{ UCHARACTERS }
```

```
VCR
Compact_Disc
!
```

This example produces the following report:

```
                Chicago Sales Actual
                Qtr1    Qtr2    Qtr3    Qtr4    Year
                =====
Television      4,410    4,001    4,934    6,261    19,606
=====
VCR             3,879    3,579    4,276    4,877    16,611
Compact_Disc    3,150    3,021    3,032    3,974    13,177
```

See Also

- [UCOLUMNS](#)
- [UDATA](#)
- [UNDERLINECHAR](#)
- [UNDERSCORECHAR](#)

UCOLUMNS

Underlines all columns, including names and data, in the preceding row.

The underline width is based on column width. If *char* is provided, it is used as the underline character. Otherwise the default character "=" is used.

Syntax

```
{ UCOLUMNS [ "char" ] }
```

Parameter Description

"char" Optional. A single-byte character, enclosed in quotation marks, that creates an underline character.

Notes

- Double-byte characters are not supported.

Example

The command {UCOLUMNS} in the following report underlines all the columns in the previous row which is the Television row.

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual

      <COLUMN (Year)
      <CHILDREN Year

<ROW (Product)
Television

{ UCOLUMNS }

VCR
Compact_Disc
!
```

This example produces the following report:

```
                Chicago Sales Actual

                Qtr1  Qtr2  Qtr3  Qtr4  Year
                =====
Television      4,410  4,001  4,934  6,261  19,606
=====
VCR             3,879  3,579  4,276  4,877  16,611
Compact_Disc    3,150  3,021  3,032  3,974  13,177
```

See Also

- [UCHARACTERS](#)
- [UDATA](#)
- [UNDERLINECHAR](#)

UDA

Selects and reports on members based on a common attribute, defined as a UDA (user-defined attribute).

Syntax

```
<UDA (dimName, udaStr)
```

Parameter Description

dimName The dimension associated with the *udaStr*.

udaStr Name of the user-defined attribute.

Notes

- If a UDA is associated with shared members, only the first instance is returned. If you want to include all instances, use the [SUDA](#) command.
- You can use the <UDA command as a standalone command or as a selection command inside the [LINK](#) statement.
- You cannot use attributes as arguments.

Example

The following example selects products that are sweet:

```
<UDA (product, "Sweet")
```

The following example uses the UDA command within a LINK statement to select level 0 products that are sweet:

```
<LINK (<UDA (product, "Sweet") AND <LEV (product, 0))
```

Note: If the Product dimension includes shared members with the UDA "Sweet", this command selects only the first instance in the outline of the shared member.

See Also

- [SUDA](#)

UDATA

Underlines data columns for a row, while not underlining the row name columns.

The underline width is based on column width. If *char* is provided, it is used as the underline character. Otherwise, the default underline character is "=".

Syntax

```
{ UDATA [ "char" ] }
```

Parameter Description

"char" Optional. A single-byte character, enclosed in quotation marks, used as the underline character.

Notes

- Double-byte characters are not supported.

Example

The command {UDATA} in the following report underlines all the data in the previous row which is the Television row.

```
<PAGE (Market, Accounts, Scenario)
```

```
Chicago Sales Actual
```

```
    <COLUMN (Year)
```

```

    <CHILDREN Year

<ROW (Product)
Television
{ UDATA }
VCR
Compact_Disc
    !

```

This example produces the following report:

```

                Chicago Sales Actual

                Qtr1   Qtr2   Qtr3   Qtr4   Year
                =====
Television      4,410   4,001   4,934   6,261   19,606
                =====
VCR             3,879   3,579   4,276   4,877   16,611
Compact_Disc    3,150   3,021   3,032   3,974   13,177

```

See Also

- [UCHARACTERS](#)
- [UNDERLINECHAR](#)

UNAME

Underlines the row name columns in the preceding row while not underlining the data columns.

If *char* is provided, then it will be used as the underline character. Otherwise, the default underline character is "=".

Syntax

```
{ UNAME [ "char" ] }
```

Parameter Description

"char" Optional. A single-byte character, enclosed in quotation marks, used as the underline character.

Notes

- Double-byte characters are not supported.

Example

The command { UNAME } in the following report underlines the row member names in the previous row which is the Television row.

```

<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual

                <COLUMN (Year)
                <CHILDREN Year

```

```
<ROW (Product)
```

```
Television  
{ UNAME }
```

```
VCR  
Compact_Disc  
!
```

This example produces the following report:

```
                Chicago Sales Actual  
  
                Qtr1   Qtr2   Qtr3   Qtr4   Year  
                ===== ===== ===== ===== =====  
Television      4,410   4,001   4,934   6,261   19,606  
=====
```

	Qtr1	Qtr2	Qtr3	Qtr4	Year
Television	4,410	4,001	4,934	6,261	19,606
VCR	3,879	3,579	4,276	4,877	16,611
Compact_Disc	3,150	3,021	3,032	3,974	13,177

See Also

- [UCHARACTERS](#)
- [UDATA](#)

UNAMEONDIMENSION

Underlines the row member names in a row whenever a member from the same dimension as the specified member changes.

Syntax

```
{ UNAMEONDIMENSION mbrName }
```

Parameter Description

mbrName Single member representing a dimension. When a new member from this dimension is output, an underline appears under all row names in the previous line.

Notes

With the ROW command, you can display members from several dimensions in columns on the side of the report. At least one member changes from one of these dimensions for each row of the report. A single report can have several UNAMEONDIMENSION commands to underline row member names, based on different dimensions which change. When combined with UNAMEONDIMENSION and PAGEONDIMENSION, UNAMEONDIMENSION is processed first, followed by SKIPONDIMENSION and PAGEONDIMENSION in order.

Example

The following example is based on Demo Basic.

```
<PAGE (Market, Accounts)  
Chicago Sales
```



```

        <COLUMN (Scenario)
        Actual
<ROW (Year, Product)
{ UNAMEONDIMENSION Year }
<CHILDREN Year
<CHILDREN Audio
    !

```

This example produces the following report:

```

=====
                        Chicago Sales Actual

Qtr1          Stereo          2,591
              Compact_Disc    3,150
              Audio           5,741
=====
Qtr2          Stereo          2,476
              Compact_Disc    3,021
              Audio           5,497
=====
Qtr3          Stereo          2,567
              Compact_Disc    3,032
              Audio           5,599
=====
Qtr4          Stereo          3,035
              Compact_Disc    3,974
              Audio           7,009
=====
Year          Stereo          10,669
              Compact_Disc    13,177
              Audio           23,846

```

See Also

- [NOPAGEONDIMENSION](#)
- [NOSKIPONDIMENSION](#)
- [PAGEONDIMENSION](#)
- [SKIPONDIMENSION](#)

UNDERLINECHAR

Sets the default underline character displayed in a report.

You can use any graphic character that you can generate in the text editor used to define the report. In some editing tools, you can generate a graphic underline by holding the ALT key down while typing 196 on the numeric keypad and then releasing the ALT key. For a double graphic underline, type 205. You must use a font with these graphic characters if the report is to print correctly. Default underline character "=" is used.

Syntax

```
{ UNDERLINECHAR [ "character" ] }
```

Parameter Description

"character" A single-byte character, enclosed in quotation marks, for the new underline character.

Notes

- Double-byte characters are not supported.

Example

```
{ UNDERLINECHAR "- " }
```

sets the character used when underlining to a single dash.

See Also

- [UCHARACTERS](#)
- [COLUMN](#)
- [UDATA](#)

UNDERSCORECHAR

Replaces the _ (underscore) character in a member name with another character.

Reports generated with this command may not be suitable for reloading into the database as report format files. Member names may no longer match the outline if the underscores are replaced.

Syntax

```
{ UNDERSCORECHAR "char" }
```

Parameter Description

"char" Single character, enclosed in quotation marks, that displays in place of underscore.

Notes

UNDERSCORECHAR is a setting command.

Example

```
{ UNDERSCORECHAR " " }
```

replaces all underscores with spaces (for example, member name New_York would appear as New York in the final report.)

WIDTH

Specifies the width of columns in a report.

If the WIDTH command is followed by *number* with no column selections, *number* sets the width for all data columns. Otherwise, the width is set for each data column listed in the command. Column numbers are assigned starting at 0 for the first row-name column,

incrementing by one for each row-name column, data column, and calculated column, in that order. The tilde character (~) follows member names or values that must be truncated to fit in the column to indicate part of the name or value is not displayed. If possible, space from adjacent columns is used to avoid truncating. The widths of names columns may be adjusted if their column numbers (0,1,...) are specifically included in the command. Alternatively, the NAMEWIDTH command may be used.

If the WIDTH command is not used, columns are wide enough to fit the widest value.

Syntax

```
{ WIDTH number [ column1 [ column2 [ columnN ] ] ] }
```

Parameter	Description
number	New column width in characters.
column1column2columnN	Optional. Numbers designating the columns to resize, separated by spaces. Values: between 0 and 161, where 0 is the first row-name column. If column-numbers are not specified, all columns are resized to the width indicated by <i>number</i> .

Notes

- The value of *n* must be zero or a positive integer.
- WIDTH is a column formatting command. If you specify columns in the WIDTH command *before* designating them with a member selection, Essbase expands the report to that number of columns. See the information on "Column Formatting Commands".
- After members for the report specification are selected, the numbers specified should not exceed the number of *columnN*.

Example

The following example is based on Sample Basic.

```
<PAGE (Measures, Market)
Illinois Sales
<SYM
{WIDTH 7}
{WIDTH 20 0}
  <COLUMN (Scenario, Year)
  Actual Budget Scenario
  Jan Feb Mar
<DESCENDANTS "100"
!
```

Which resizes all data columns to a WIDTH of seven and the row name label column (column 0) to a WIDTH of 20.

Sales Illinois									
	Actual			Budget			Scenario		
	Jan	Feb	Mar	Jan	Feb	Mar	Jan	Feb	Mar
	=====	=====	=====	=====	=====	=====	=====	=====	=====
100-10	345	354	367	360	370	380	345	354	367

```
100-20          234    254    267    240    260    280    234    254    267
100-30          #Missi #Missi #Missi #Missi #Missi #Missi #Missi #Missi #Missi
```

See Also

- [NAMEWIDTH](#)

WITHATTR

Specifies the characteristics of a base-dimension member that match the specified values in a report script. You must create attribute dimensions in the outline and associate them with a base dimension before you use WITHATTR.

Syntax

```
<WITHATTR (dimName, "operator", value)
```

Parameter Description

dimName Single attribute dimension name.

"operator" Operator specification, which must be enclosed in double quotes ("").

The supported operators are:

- > (Greater than)
- >= (Greater than or equal to)
- < (Less than)
- <= (Less than or equal to)
- = = (Equal to)
- <> or != (Not equal to)
- IN (Within a specified range)

Note: These operators may behave differently depending on the attribute type with which you use them. See the table in Examples for more information.

value Value that, in combination with the operator, defines the condition that must be met. The *value* can be an attribute member specification, a constant, or a date-format function (for example, <TODATE).

Notes

This command specifies two or more attribute dimension tags, which are associated with a base dimension. If you use the <WITHATTR syntax, the command is applied only to a specific query.

Example

Example 1

The following table shows examples, based on the Sample Basic database, for each type of operator:

Operator	Example	Result
>	<WITHATTR(Population,">","18000000")	Returns New York, California, and Texas
>=	<WITHATTR(Population,">=",10000000) where 10,000,000 is not a numeric attribute member, but a constant	Returns New York, Florida, California, Texas, Illinois, and Ohio
<	<WITHATTR(Ounces,"<","16")	Returns Cola, Diet Cola, Old Fashioned, Sasparilla, and Diet Cream
<=	<WITHATTR("Intro Date","<=",<TODATE("mm-dd-yyyy", "04-01-1996"))	Returns Cola, Diet Cola, Caffeine Free Cola, and Old Fashioned
= =	<WITHATTR("Pkg Type","= =",Can)	Returns Cola, Diet Cola, and Diet Cream
<> or !=	<WITHATTR(Caffeinated,"<>","True)	Returns Caffeine Free Cola, Sasparilla, Birch Beer, Grape, Orange, Strawberry
IN	<WITHATTR("Population","IN",Medium)	Returns Massachusetts, Florida, Illinois, and Ohio

Example 2

The following report script

```
<PAGE (Product, Measures, Scenario)
Florida Sales Actual

<COLUMN (Year)
<CHILDREN Year

<ROW (Market)
<WITHATTR(Population IN Large)
!
```

returns on rows only those members of Market whose Population attributes map to ranges defined as Large:

Product Sales Actual					
	Qtr1	Qtr2	Qtr3	Qtr4	Year
	=====	=====	=====	=====	=====
New York	7,705	9,085	9,325	8,583	34,698
California	11,056	12,164	13,073	11,149	47,442
Texas	4,505	4,589	4,807	4,402	18,303

See Also

- [<ATTRIBUTE](#)
- [<TODATE](#)

WITHATTREX

Specifies the characteristics of a base-dimension member that match the specified values in a report script. You must create varying attribute dimensions in the outline and associate them with a base dimension before you use WITHATTREX in a report script.

Syntax

`<WITHATTREX (dimName, "operator", value, options, startTuple[, endTuple])`

Parameter	Description
dimName	Single varying attribute dimension name.
"operator"	Operator specification, which must be enclosed in double quotes (""). The supported operators are: <ul style="list-style-type: none">● > (Greater than)● >= (Greater than or equal to)● < (Less than)● <= (Less than or equal to)● = = (Equal to)● <> or != (Not equal to)● IN (Within a specified range)
value	Value that, in combination with the operator, defines the condition that must be met. The <i>value</i> can be an attribute member specification, a constant, or a date-format function (for example, <TODATE).
options	ANY
startTuple[, endTuple]	(m1, m2, . . . , mN) Level-0 members from one or more independent dimensions for attributeMemberName may be part of the input tuple. Members from all independent dimensions should be listed. If a member is not listed, the member of the same dimension from the current query or calculation context is used.

Notes

This command specifies two or more attribute dimension tags, which are associated with a base dimension. If you use the <WITHATTREX syntax, the command is applied only to a specific query.

Example

```
<withattrex("intro date", "<=", <today("mm-dd-yyyy", "04-01-1996"), ANY, (jan), (jun))
```

```
<withattrex(ounces, ">", "16", ANY, (jan), (jun))
```

See Also

- [<ATTRIBUTEVA](#)
- [<PERSPECTIVE](#)
- [<TODATE](#)

ZEROTEXT

Replaces zero data values with a text string if a zero data value is output.

Syntax

```
{ ZEROTEXT [ "text" ] }
```

Parameter Description

text Optional. String, in quotation marks, to use in place of 0.

Notes

All data values less than .00000000000001 and greater than -.00000000000001 are treated as 0, as well as all data values that would be displayed as 0, regardless of their true value.

Default Value

If you do not specify text, the default 0 is restored.

Example

```
{ZEROTEXT "-"}
```

changes a 0 value to -.

See Also

- [MISSINGTEXT](#)

In This Chapter

Essbase Unicode File Utility Overview	1377
Types of Encoding Indicators	1378
Determining Whether to Use UTF-8 or Non-Unicode Text Files	1378
When to Use the Essbase Unicode File Utility.....	1379
Essbase Unicode File Utility Syntax	1379

Essbase Unicode File Utility Overview

The Essbase Unicode File Utility is a standalone program that enables you to add encoding identifiers to files used with Unicode-mode applications. Encoding identifiers are markers that identify the text encoding used in the file. Located in the *ESSBASEPATH*\bin directory, this utility is called *essutf8.exe* (in Windows) or *ESSUTF8* (in UNIX). You can use this utility to make the following changes to text files, outline files, and rules files:

- Add a UTF-8 signature to UTF-8-encoded text files
- Convert non-Unicode-encoded text files to UTF-8 encoding, including the UTF-8 signature
- Insert a locale indicator in non-Unicode-encoded files, including script files, data source files, outline files (.otl) and rules files (.rul)
- Remove locale indicators from non-Unicode-encoded files
- Backup the files before changing them

The Essbase Unicode File Utility works with text files and binary files that you can edit and change. This utility does not support user-defined characters (UDC) such as can be found in Japanese, Korean, Chinese, and Taiwanese host code pages.

Applicable text files include:

- Calculation scripts (.csc)
- Report scripts (.rep)
- Data source files for dimension builds, data loads, and partition area definitions
- Alias table import files (.alt)

Applicable binary files include:

- Outline files (.otl)

- Rules files (.rul)

Using the Essbase Unicode File Utility to insert locale indicators in outline files and rules files is relevant when outline files and rules files were created by earlier releases of Essbase or its clients (prior to Release 7.0) or when rules files are initially created on a client. For a more detailed description of encoding and locale indicators, see the "Enabling Multi-Language Applications Through Unicode" part in the *Oracle Essbase Database Administrator's Guide*.

Note: Text files for non-Unicode-mode applications cannot be encoded in UTF-8. They must be encoded according to a locale definition common to the client and Essbase Server.

See the "Enabling Multi-language Applications through Unicode" part of the *Oracle Essbase Database Administrator's Guide* for additional information about encoding formats, the UTF-8 signature, and locale indicators.

Types of Encoding Indicators

Different types of encoding indicators are used, depending on the type of file and its encoding:

- The *UTF-8 signature*, which indicates that a text file is encoded in UTF-8, is a mark at the beginning of the file. Although optional within the computer industry, Essbase requires that UTF-8-encoded files include the UTF-8 signature.
- Inserted at the beginning of non-Unicode-encoded text files, the *locale header record* is an additional text record that includes a locale that identifies the encoding of the file. You can use the Essbase Unicode File Utility to insert the locale header or you can use a text editor to create the locale header. For the format and other details about the locale header record, see the *Oracle Essbase Database Administrator's Guide*.
- As binary files that contain text information, outline files and rules files contain a flag that indicates whether the text is encoded in UTF-8 or in a supported non-Unicode encoding.
- As needed, if a file is not UTF-8-encoded, Essbase uses an internal locale indicator to identify the locale used for character text encoding.

Determining Whether to Use UTF-8 or Non-Unicode Text Files

While you are migrating various client and server sites to a Unicode-enabled release with Unicode-mode applications, Essbase provides you the flexibility to use non-Unicode-encoded files. For Unicode-mode applications, using UTF-8-encoded text files is recommended. Using UTF-8 encoding is simpler; you do not need to keep track of different locales.

When to Use the Essbase Unicode File Utility

The following list includes examples of situations when you would use the Essbase Unicode File Utility.

- To determine if a file contains an encoding indicator and, if it does, how it is encoded.
- To add a UTF-8 signature to a UTF-8-encoded file. UTF-8-encoded files must include the UTF-8 signature.
- To add a locale indicator to an outline file or rules file that is input to a Unicode-mode application on a Release 7.0 Essbase Server, if the file was created by an earlier release of Essbase.
- To remove a locale indicator from a file created by Release 7.0 Oracle Essbase Administration Services, if the file is to be used with an Essbase Server release prior to 7.0.

Note: Release 7.0 rules files that are not compatible with prior releases of Essbase are excepted.

For a more detailed description of encoding and locale indicators, see the *Oracle Essbase Database Administrator's Guide*.

Essbase Unicode File Utility Syntax

The Essbase Unicode File Utility (`ESSUTF8` or `essutf8`) modifies files to be used with Unicode-mode applications. Use this utility to make the following changes to files:

- Add a UTF-8 signature to UTF-8-encoded text files
- Convert non-Unicode-encoded text files to UTF-8 encoding, including the UTF-8 signature
- Insert a locale indicator in non-Unicode-encoded files including script files, data sources, outline files (`.otl`) and rules files (`.rul`)
- Remove locale indicators from non-Unicode-encoded files
- Backup files before changing them

For a description of encoding indicators, see the [“Types of Encoding Indicators” on page 1378](#).

```
essutf8 [option] filespec
```

Syntax	Description
[option]	Case-sensitive, lowercase execution options. A single command can include more than one execution option. Include the hyphen at the beginning of each execution option within a command. See Table 2, “Execution Options,” on page 1380 . The <code>-c</code> , <code>-d</code> , <code>-i</code> , and <code>-s</code> options may not be used in combination. The remaining options may be used in combination with one of the four options, or in combination with each other. See Notes for more information.

Syntax	Description
<i>filespec</i>	<p>Location and names of files. You can specify any of the following items:</p> <ul style="list-style-type: none"> ● A file name in the current directory ● An absolute path that includes the file name ● A file-name mask containing the * (asterisk) and ? (question mark) wildcards (for example, <code>abc*.txt</code> includes all files with names starting with <code>abc</code> and ending with the extension <code>.txt</code>) <p>Caution! To avoid corruption of binary files not related to Oracle Essbase, do not use wildcards within file extensions (for example, do not specify anything like <code>xyz.*</code> or <code>*.*</code>). Use of wildcards is recommended only within the portion of the file name before the dot; for example, <code>*.scr</code> or <code>*2002.txt</code>.</p>

Table 2 Execution Options

Option	Description
-a	Lists supported locales. You can copy locales from the list into the clipboard.
-b	Creates a backup file (<code>.bak</code>) for each modified file.
-c	<p>Converts text files without a UTF-8 signature to UTF-8 encoding, removing existing locale indicators and inserting a UTF-8 signature in each file.</p> <p>Caution! The utility cannot recognize a file to be in UTF-8 encoding if the file does not contain a UTF-8 signature. Be sure to use the <code>-c</code> option only with files that are not in UTF-8 encoding. Using the <code>-c</code> option with files that are in UTF-8 encoding results in files that are not usable. To add a UTF-8 signature to a UTF-8 encoded file, use the <code>-s</code> option as described below.</p>
-d	Deletes locale indicators from specified non-Unicode outline and rules files.
-e	Displays the encoding of each specified text, outline, or rules file.
-h	Displays help text. This is the default option.
-i	Inserts a locale indicator in each non-Unicode file that does not have an indicator. If a <code>-l</code> option is not included to specify a locale, the <code>ESSLANG</code> locale is assumed.
-l <i>locale</i>	<p>Specifies the locale for the locale indicator for a non-Unicode file. For <i>locale</i>, use the following locale format:</p> <pre><language>_<territory>.<code page name>@<sort sequence></pre> <p>Supported locales are listed in the <i>Oracle Essbase Database Administrator's Guide</i></p> <p>Caution! Do not add a locale indicator to a file containing UTF-8 encoding.</p>
-q	Defines a quiet operation. No messages are displayed.
-s	<p>Adds a UTF-8 signature to each text file that does not have a UTF-8 signature or a locale header</p> <p>Caution! The utility cannot recognize a file to be in UTF-8 encoding if the file does not contain a UTF-8 signature. Be sure to use the <code>-s</code> option only with files that are not in UTF-8 encoding. Using the <code>-s</code> option with files that are in UTF-8 encoding results in files that are not usable.</p>

Notes

- In Windows, run `essutf8.exe`; in UNIX, run `ESSUTF8`.
- Backing up files (option `-b`) is recommended.

- To process UTF-8 encoded files, Essbase Server requires the files include the UTF-8 signature.
- Do not combine a UTF-8 signature and locale header in the same file. If a text file contains both types of encoding indicators, the file is interpreted as UTF-8 encoded, and the locale header is read as the first data record.
- See the "Enabling Multi-Language Applications Through Unicode" part in the *Oracle Essbase Database Administrator's Guide* for more information about file encoding.
- Ensure that the encoding and condition of the specified files is what the specified operation expects. For example, do not define a command to delete locale indicators from non-Unicode-encoded files that do not contain locale indicators.

Examples

Backup plus UTF-8 signature insertion

```
essutf8 -b -s salesjune.utf8
```

Backup plus insertion of locale header record

```
essutf8 -b -i -l Spanish_Spain.Latin1@Spanish complex.rep
```

Backup plus conversion of multiple files to UTF-8 encoding

```
essutf8 -b -c *.txt
```

Backup plus deletion of locale indicator in a rules file

```
essutf8 -b -d \EssbaseServer\app\demo\basic\genref.rul
```


Index

Symbols

! (bang).. *See* bang
 "" "" quotation marks
 in MaxL Shell, 828
 %. *See* Operators
 & (ampersand).. *See* ampersand
 *. *See* Operators
 +. *See* Operators
 -. *See* Operators
 /. *See* Operators
 ; semicolon
 in MaxL statements, 828
 <. *See* Operators
 < left angle bracket (in reports), 1170
 <=. *See* Operators
 <>. *See* Operators
 =. *See* Operators
 >. *See* Operators
 >=. *See* Operators
 { } curly braces (in reports), 1170

A

ABS
 calculation function, 39
 MDX function, 998
 absolute value function
 in calculation scripts or formulas, 39
 in MDX queries, 998
 ACCOFF report writer command, 1216
 ACCON report writer command, 1217
 accounts tag
 calculation function returning TRUE if present,
 105
 MDX function returning TRUE if present, 1075
 ACCUM, 40
 ACTION specification for triggers, 767
 AFTER report writer command, 1218

AGENTDELAY configuration setting, 382
 AGENTDESC configuration setting, 383
 AGENTDISPLAYMESSAGELEVEL configuration
 setting, 383
 AGENTLEASEEXPIRATIONTIME configuration
 setting, 384
 AGENTLEASEMAXRETRYCOUNT configuration
 setting, 385
 AGENTLEASERENEWALTIME configuration
 setting, 385
 AGENTLOGMESSAGELEVEL configuration setting,
 386
 AGENTPORT configuration setting, 387
 AGENTSECUREPORT configuration setting, 388
 AGENTTHREADS configuration setting, 388
 AGG calculation command, 309
 aggregate storage databases
 clearing data by region, 874
 creating an application, 882
 creating outline formulas, 973
 data load buffers, using, 923
 list aso_level_info, 904
 list data load buffers, 904
 MaxL data load process, 921
 MaxL statements enabled for, 864
 merging data slices, 873
 outline optimization settings, 480, 483, 484
 overview, 19
 tablespaces, 642
 aggregations
 performing using MaxL, 738
 AGGRESSIVEBLKOPTIMIZATION configuration
 setting, 389
 AGTMAXLOGFILESIZE configuration setting, 390
 AGTSVRCONNECTIONS configuration setting, 391
 algorithms
 for data mining
 creating, 910

- deleting, [918](#)
- displaying, [915](#)
- for the @Trend calculation function, [210](#)
- DES, [21](#)
- linear regression, [21](#)
- linear regression with seasonal adjustment, [21](#)
- SES, [21](#)
- TES, [21](#)
- ALIAS calculation function, [42](#)
- aliases
 - in reports, [1171](#)
 - managing in ESSCMD, [534](#)
 - managing in MaxL, [649](#), [653](#)
- ALLANCESTORS calculation function, [41](#)
- ALLINSAMEDIM report writer command, [1219](#)
- ALLOCATE calculation function, [42](#)
- allocation functions, [27](#)
- allocations
 - executing in MaxL, [886](#)
- ALLSIBLINGS report writer command, [1220](#)
- alter (MaxL), [633](#)
- ampersand
 - calculation command, [309](#)
 - report writer command, [1216](#)
- Ancest calculation function, [46](#)
- ancestors
 - calculation function that returns an ancestor, [46](#)
 - calculation function that tests for an ancestor, [106](#)
 - calculation functions that return ancestor values, [48](#), [140](#), [185](#)
 - calculation functions that return more than an ancestor, [41](#)
 - MDX function that returns one ancestor, [1000](#)
 - MDX function that returns set of ancestors, [1001](#)
 - MDX function that tests for an ancestor, [1075](#)
 - report command that returns ancestors, [1221](#)
- ancestors calculation function, [47](#)
- Ancestors functions
 - MDX function, [1001](#)
 - report writer command, [1221](#)
- Ancestval calculation function, [48](#)
- AND
 - calculation operator, [24](#)
 - MDX operator, [963](#)
- APP-NAME in MaxL, [767](#)
- applications
 - creating in MaxL, [680](#)
 - aggregate storage applications, [882](#)
 - deleting in MaxL, [730](#)
 - displaying in MaxL, [707](#)
 - modifying in MaxL, [642](#)
 - sample applications used in this document, [17](#)
- APPMAXLOGFILESIZE configuration setting, [392](#)
- APSRESOLVER configuration setting, [392](#)
- ARRAY calculation command, [310](#)
- array variables, [310](#)
- ASOLOADBUFFERWAIT configuration setting, [393](#)
- ASOSAMPLESIZEPERCENT configuration setting, [394](#)
- ASYM report writer command, [1222](#)
- Attribute function
 - in calculation scripts or formulas, [49](#)
 - in MDX queries, [1002](#)
- ATTRIBUTE report writer command, [1223](#)
- Attribute-value calculation functions
 - for Boolean values, [50](#)
 - for numeric or date values, [53](#)
 - for text values, [51](#)
- Attributebval calculation function, [50](#)
- AttributeEx function
 - in MDX queries, [1003](#)
- attributes
 - as MDX properties, [964](#)
 - calculation function that returns base members of attributes, [106](#)
 - calculation function that returns members with attribute, [49](#)
 - calculation function that returns members with type of attribute, [225](#)
 - MDX function that returns members with attribute, [1002](#)
 - MDX function that returns members with type of attribute, [1157](#)
- Attributesval calculation function, [51](#)
- ATTRIBUTEVA report writer command, [1224](#)
- Attributeval calculation function, [53](#)
- AUTHENTICATIONMODULE configuration setting, [395](#)
- Average function
 - in calculation scripts or formulas, [54](#)
 - in MDX queries, [1004](#)
- averages
 - Avg MDX function, [1004](#)
 - calculation functions, [54](#), [55](#), [153](#), [311](#)

Avgrange calculation function, 55
axis specification in MDX queries, 949

B

bang command (report writer), 1216
BEFORE report writer command, 1225
between calculation function, 56
block locks
 limiting number of locks, 401
 limiting the time a lock is held, 464
block storage databases
 outline optimization settings, 484
BLOCKHEADERS report writer command, 1226
blocks
 calculating updated, 366
 clearing/deleting, 316
 locking to calculate, 359
 preventing creation of during calculation, 345, 346
Boolean functions
 in calculator, 27
 in MDX, 989
BOTTOM report writer command, 1227
bottom-up calculations, 358
BottomCount MDX function, 1006
BottomPercent MDX function, 1007
BottomSum MDX function, 1009
BRACKETS report writer command, 1230
buffer
 aggregate storage data loads, 921
 data load buffer, 923
buffers
 list of, 904

C

caches
 aggregate storage formula cache, 467
 calculator cache, setting size, 337
CALC ALL calculation command, 311
CALC AVERAGE calculation command, 311
CALC DIM calculation command, 312
CALC FIRST calculation command, 313
CALC LAST calculation command, 313
CALC TWOPASS calculation command, 314
CALC-NAME in MaxL, 767
CALC-STRING in MaxL, 767
CALCCACHE configuration setting, 396

CALCCACHEDEFAULT configuration setting, 398
CALCCACHEHIGH configuration setting, 397
CALCCACHELOW configuration setting, 399
CALCLIMITFORMULARECURSION configuration setting, 400
CALCLOCKBLOCK configuration setting, 401
Calcmode calculation function, 57
CALCMODE configuration setting, 402
CALCNOTICE configuration setting, 403
CALCOPTFRMBOTTOMUP configuration setting, 404
CALCPARALLEL configuration setting, 406
CALCREUSEDYNCALCBLOCKS configuration setting, 405
CALCTASKDIMS configuration setting, 407
CALCULATE COLUMN report writer command, 1230
CALCULATE ROW report writer command, 1233
calculated members in MDX, 954
calculation commands
 groups of, 305
 overview of, 303
calculation operators
 in calculator syntax, 24
 in MDX syntax, 963
calculations
 average members, 311
 clean data blocks and, 341
 columns and rows in reports, 1171
 commands for, 303
 creating in MaxL, 681
 currency conversions, 314
 custom, executing in MaxL, 890
 deleting in MaxL, 730
 dirty blocks, 366
 displaying in MaxL, 709
 entire database, 311
 executing in MaxL, 737
 first members, 313
 formulas and aggregations, 312
 ignoring member formulas, 309
 iterative, 335
 last members, 313
 locking blocks concurrently, 359
 monitoring progress, 360
 optimizing formulas on sparse dimensions, 358
 overview of commands, 303

- parallel, [338](#)
- parents with same defined currency, [367](#)
- restricting to database subset, [331](#)
- rows and columns in reports, [1171](#)
- sparse dimensions, [309](#)
- temporary variables, [368](#)
- two-pass members, [314](#)
- types of, [305](#)
- varying attributes perspective command, [365](#)
- calculator cache, [337](#)
- CASE MDX function, [1010](#)
- CCONV calculation command, [314](#)
- CCTRACK configuration setting, [408](#)
- CellValue MDX function, [1013](#)
- characters
 - underlining in reports, [1363](#)
- characters in MaxL, [767](#)
- Children function
 - in calculation scripts or formulas, [64](#)
 - in MDX queries, [1015](#)
- CHILDREN report writer command, [1236](#)
- clean status, [341](#)
- CLEARALLROWCALC report writer command, [1236](#)
- CLEARBLOCK calculation command, [316](#)
- CLEARCCTRACK calculation command, [317](#)
- CLEARDATA calculation command, [317](#)
- clearing
 - blocks, [316](#)
 - data, [317](#)
 - data by region (aggregate storage), [874](#)
 - exchange rate tracking (CCTRACK), [317](#)
- CLEARLOGFILE configuration setting, [410](#)
- CLEARROWCALC report writer command, [1237](#)
- CLIENTPREFERREDMODE configuration setting, [411](#)
- ClosingPeriod MDX function, [1016](#)
- clusters
 - in MDX, [971](#)
- cmd2mxl utility, [848](#)
- CoalesceEmpty MDX function, [1018](#)
- COLHEADING report writer command, [1237](#)
- column or row calculation commands, [1171](#)
- COLUMN report writer command, [1239](#)
- columns
 - calculating, [1230](#)
 - displaying dimensions as, [1230](#)
 - underlining in a report, [1364](#)
- commands
 - calculations, [303](#)
- COMMAS report writer command, [1239](#)
- COMMENT-STRING in MaxL, [767](#)
- comments in MDX, [970](#)
- compaction of security file, [491](#)
- Compound calculation function, [65](#)
- Compoundgrowth calculation function, [66](#)
- Concatenate calculation function, [66](#)
- CONDITION specification for triggers, [767](#)
- conditional calculation commands, [305](#)
- conditional functions
 - in MDX, [1010](#), [1047](#), [1057](#), [1069](#)
- conditional tests, [305](#)
- consolidating
 - #MISSING values, [336](#)
 - only parents with same currency, [367](#)
- constraints on data, [699](#), [927](#)
- control flow calculation commands, [305](#)
- copying data, [318](#)
- Correlation calculation function, [67](#)
- Count function
 - in calculation scripts or formulas, [69](#)
 - in MDX queries, [1019](#)
- count functions in MDX queries, [1006](#), [1019](#), [1111](#), [1147](#)
- Cousin MDX function, [1020](#)
- CRASHDUMP configuration setting, [411](#)
- create (MaxL), [633](#)
- create blocks on equation, [346](#)
- CrossJoin MDX function, [1022](#)
- cube
 - deploying, [703](#), [840](#)
- CUBE AREA specification for triggers, [767](#)
- cube specification in MDX queries, [952](#)
- Curgen calculation function, [70](#)
- CURHEADING report writer command, [1240](#)
- Curlev calculation function, [72](#)
- currency conversions
 - calculating, [314](#)
 - displaying headings of, [1240](#)
 - performing in a report, [1240](#)
 - suppressing information in reports, [1344](#)
 - tracking, [341](#)
- currency members, [367](#)
- CURRENCY report writer command, [1240](#)

- CurrentMember MDX function, [1024](#)
 - CurrentTuple MDX function, [1025](#)
 - Currnbr calculation function, [73](#)
 - Currnbrange calculation function, [74](#)
 - configuring behavior of, [490](#)
 - custom-defined functions
 - creating in MaxL, [686](#)
 - deleting in MaxL, [732](#)
 - displaying in MaxL, [713](#)
 - overview, [234](#)
 - custom-defined macros
 - creating in MaxL, [689](#)
 - deleting in MaxL, [734](#)
 - displaying in MaxL, [716](#)
 - overview, [292](#)
- D**
- data
 - appearance in reports, [1171](#)
 - clearing sections of, [317](#)
 - copying, [318](#)
 - exporting using MaxL, [743](#)
 - importing using MaxL, [752](#)
 - ordering in reports, [1291](#)
 - restricting ranges in reports, [1171](#)
 - data blocks.. *See* blocks
 - data constraints, [699](#), [927](#)
 - data declaration calculation commands, [305](#)
 - data load buffer, [921](#)
 - data mining, [907](#)
 - data ordering commands, [1171](#)
 - data range commands, [1171](#)
 - data slices (aggregate storage)
 - merging, [873](#)
 - database objects
 - deleting in MaxL, [734](#)
 - displaying in MaxL, [717](#)
 - modifying in MaxL, [663](#)
 - databases
 - calculating, [303](#)
 - clearing data from, [317](#)
 - consolidating parent/child relationships only, [309](#)
 - creating in MaxL, [682](#)
 - deleting in MaxL, [730](#)
 - displaying in MaxL, [709](#)
 - mining using algorithms, [907](#)
 - modifying in MaxL, [646](#), [653](#)
 - modifying settings in MaxL, [649](#)
 - DATA CACHESIZE configuration setting, [412](#)
 - DATA COPY calculation command, [318](#)
 - DATA ERROR LIMIT configuration setting, [413](#)
 - DATA EXPORT calculation command, [319](#)
 - DATA EXPORT COND calculation command, [323](#)
 - DATA EXPORT ENABLE BATCH INSERT configuration setting, [414](#)
 - DATA FILE CACHESIZE configuration setting, [415](#)
 - DATA IMPORT BIN calculation command, [325](#)
 - date
 - get first day in MDX queries, [1060](#)
 - get last day in MDX queries, [1062](#)
 - get next day in MDX queries, [1063](#)
 - get rounded date in MDX queries, [1063](#)
 - date (Julian)
 - in MDX queries, [1086](#)
 - date (UNIX)
 - in MDX queries, [1154](#)
 - DATE FORMAT report writer command, [1241](#)
 - DBS-NAME in MaxL, [767](#)
 - DECIMAL report writer command, [1242](#)
 - Decline calculation function, [80](#)
 - DEFAULT LOG LOCATION configuration setting, [415](#)
 - DefaultMember MDX function, [1031](#)
 - defragmentation of security file, [491](#)
 - DELAYED RECOVERY configuration setting, [416](#)
 - DELIMITED MSG configuration setting, [417](#)
 - DELIMITER configuration setting, [417](#)
 - delimiters
 - in reports, [1170](#)
 - deploying cube from Essbase Studio, [703](#), [840](#)
 - Descendants function
 - in MDX queries, [1032](#)
 - descendants function
 - in calculation scripts or formulas, [81](#)
 - DESCENDANTS report writer command, [1243](#)
 - DEXPSQL ROW SIZE configuration setting, [418](#)
 - DIM BOTTOM report writer command, [1245](#)
 - DIM BUILD ERROR LIMIT configuration setting, [419](#)
 - DIM BUILD STATS INTERVAL configuration setting, [420](#)
 - DIM END report writer command, [1246](#)
 - Dimension MDX function, [1036](#)
 - dimensions
 - as report page members, [1301](#)

calculating, [312](#)
 calculation functions that work on multiple dimensions, [136](#), [140](#), [141](#), [142](#)
 importing using MaxL, [754](#)
 rows in reports, [1319](#)
 specifying in MDX queries, [958](#)
 DIMTOP report writer command, [1247](#)
 DIRECTIO configuration setting, [420](#)
 dirty blocks, [366](#)
 DISABLEREPLMISSINGDATA configuration setting, [421](#)
 Discount calculation function, [82](#)
 disk volumes
 adding using MaxL, [659](#)
 deleting using MaxL, [659](#)
 displaying in MaxL, [710](#)
 setting using MaxL, [659](#)
 display (MaxL), [634](#)
 DISPLAYMESSAGELEVEL configuration setting, [423](#)
 Distinct MDX function, [1035](#)
 DLSINGLETHREADPERSTAGE configuration setting, [424](#)
 DLTHREADSPREPARE configuration setting, [426](#)
 DLTHREADSWRITE configuration setting, [427](#)
 drill-through URLs
 creating in MaxL, [683](#)
 deleting in MaxL, [731](#)
 displaying in MaxL, [711](#)
 modifying in MaxL, [660](#)
 DrillDown MDX functions
 byLayer, [1036](#)
 Member, [1037](#)
 DrillUp MDX functions
 byLayer, [1039](#)
 Member, [1041](#)
 drop (MaxL), [635](#)
 DTS MDX function, [1042](#)
 DUPLICATE report writer command, [1248](#)
 DYNCALCCACHEBLKRELEASE configuration setting, [429](#)
 DYNCALCCACHEBLKTIMEOUT configuration setting, [430](#)
 DYNCALCCACHECOMPRBLKBUFSIZE configuration setting, [432](#)
 DYNCALCCACHEMAXSIZE configuration setting, [433](#)

DYNCALCCACHEONLY configuration setting, [435](#)
 DYNCALCCACHEWAITFORBLK configuration setting, [436](#)

E

ELSE calculation command, [326](#)
 ELSEIF calculation command, [327](#)
 ENABLE_DIAG_TRANSPARENT_PARTITION configuration setting, [438](#)
 ENABLECLEARMODE configuration setting, [439](#)
 ENABLESECUREMODE configuration setting, [440](#)
 ENABLESWITCHTOBACKUPFILE configuration setting, [440](#)
 encoding management, [1377](#)
 ENDHEADING report writer command, [1249](#)
 ENDIF calculation command, [328](#)
 EnumText MDX function, [1043](#)
 EnumValue function
 in calculation scripts or formulas, [83](#)
 Enumvalue MDX function, [1044](#)
 equal calculation function, [83](#)
 Essbase
 changing system properties in MaxL, [668](#)
 viewing system information in MaxL, [721](#)
 Essbase sessions
 displaying in MaxL, [720](#)
 stopping in MaxL, [668](#)
 essbase.cfg file, [369](#)
 essbase.pm, [841](#)
 ESSBASEFAILOVERTRACELEVEL configuration setting, [441](#)
 ESSBASESERVERHOSTNAME configuration setting, [442](#)
 ESSCMD, [526](#)
 batch mode, [522](#)
 batch-processing mode, [519](#)
 commands, [534](#)
 interactrve mode, [519](#)
 migrating from ESSCMD to MaxL, [848](#)
 multi-user considerations, [519](#)
 starting, [526](#)
 syntax, [520](#)
 ESSCMD commands, [534](#)
 essutf8 utility, [1377](#)
 EUROPEAN report writer command, [1250](#)
 Except MDX function, [1044](#)

EXCEPTIONLOGOVERWRITE configuration setting, [442](#)

exchange rates, [314](#), [317](#)

EXCLUDE...ENDEXCLUDE calculation command, [329](#)

EXCLUSIVECALC configuration setting, [444](#)

execute (MaxL), [635](#)

Exp (exponent) function

in calculation scripts or formulas, [84](#)

in MDX queries, [1045](#)

expand calculation function, [85](#)

export (MaxL), [636](#)

EXPORTTHREADS configuration setting, [444](#)

Extract MDX function, [1046](#)

F

Factorial function

in calculation scripts or formulas, [87](#)

in MDX queries, [1047](#)

FAILOVERMODE configuration setting, [445](#)

FEEDON report writer command, [1251](#)

file encoding indicators, [1377](#)

FILE-NAME in MaxL, [767](#)

FILELOCKINGMODE configuration setting, [446](#)

files

locking on UNIX, [446](#)

Filter MDX function, [1047](#)

FILTER-NAME in MaxL, [767](#)

filtering on metadata, [926](#)

filters

creating in MaxL, [684](#)

deleting in MaxL, [731](#)

displaying in MaxL, [712](#)

modifying in MaxL, [661](#)

financial functions, [27](#)

first members, calculating, [313](#)

FirstChild MDX function, [1053](#)

FirstSibling MDX function, [1053](#)

FIX...ENDFIX calculation command, [331](#)

FIXCOLUMNS report writer command, [1251](#)

flattened sets

in MDX, [971](#)

flow commands.. *See* control flow calculation commands

FORCEGRIDEXPANSION configuration setting, [447](#)

forecasting functions, [27](#)

format commands in report writer, [1171](#)

FORMATCOLUMNS report writer command, [1253](#)

formulas

cache size for (aggregate storage), [467](#)

ignoring during calculation, [309](#)

MDX formulas for aggregate storage databases, [973](#)

optimizing calculation of, [358](#)

fragmentation of security file, [491](#)

FROM section in MDX queries, [952](#)

FUNC-NAME in MaxL, [767](#)

functional calculation commands, [305](#)

functions (custom defined)

creating in MaxL, [686](#)

deleting in MaxL, [732](#)

displaying in MaxL, [713](#)

functions for calculator, [37](#)

G

Gen calculation function, [89](#)

GEN report writer command, [1254](#)

Generate MDX function, [1057](#)

Generation MDX function, [1058](#)

generations

calculation functions pertaining to, [70](#), [89](#), [108](#), [115](#)

described, [21](#)

in MaxL, [959](#)

MDX functions pertaining to, [1058](#), [1059](#), [1078](#)

Generations MDX function, [1059](#)

Genmbrs calculation function, [89](#)

GetFirstDay

MDX function, [1060](#)

GetLastDay

MDX function, [1062](#)

GetNextDay

MDX function, [1063](#)

GetRoundDate

MDX function, [1063](#)

grant (MaxL), [750](#)

GRIDEXPANSION configuration setting, [448](#)

GRIDEXPANSIONMESSAGES configuration setting, [448](#)

GROUP-NAME in MaxL, [767](#)

groups

creating in MaxL, [687](#)

deleting in MaxL, [732](#)

displaying in MaxL, [714](#)
 modifying in MaxL, [662](#)
 Growth calculation function, [91](#)

H

HAENABLE configuration setting, [449](#)
 HAMAXNUMCONNECTION configuration setting, [449](#)
 HAMAXNUMSQLQUERY configuration setting, [450](#)
 HAMAXQUERYROWS configuration setting, [451](#)
 HAMAXQUERYTIME configuration setting, [452](#)
 HAMEMORYCACHESIZE configuration setting, [453](#)
 HARAGGEDHIERARCHY configuration setting, [454](#)
 HARETRIEVENUMROW configuration setting, [455](#)
 HASOURCEDSNOS390 configuration setting, [456](#)
 Head MDX function, [1064](#)
 HEADING report writer command, [1256](#)
 headings of report pages.. *See* page headings
 Hierarchize MDX function, [1067](#)
 HISLEVELDRILLTHROUGH configuration setting, [457](#)
 HOST-NAME in MaxL, [767](#)
 Hybrid Analysis

- controlling memory cache size, [453](#)
- disabling in report scripts, [1256](#)
- enabling in report scripts, [1255](#)
- enabling/disabling, [449](#)
- enabling/disabling query logging, [649](#)
- handling null values, [454](#)
- limiting connections to relational source, [449](#)
- limiting number of rows, [451](#), [455](#)
- limiting number of SQL queries, [450](#)
- limiting query time, [452](#)

 HYBRIDANALYSISOFF report writer command, [1256](#)
 HYBRIDANALYSISON report writer command, [1255](#)

I

iallancestors calculation function, [91](#)
 iancestors calculation function, [93](#)
 IANCESTORS report writer command, [1257](#)
 IBHFIXTHRESHOLD configuration setting, [457](#)

Ichildren calculation function, [94](#)
 ICHILDREN report writer command, [1258](#)
 idescendants calculation function, [94](#)
 IDESCENDANTS report writer command, [1259](#)
 IF calculation command, [333](#)
 iallancestors calculation function, [96](#)
 IIF MDX function, [1069](#)
 idescendants calculation function, [97](#)
 ILSiblings calculation function, [99](#)
 IMMHEADING report writer command, [1261](#)
 IMPLIED_SHARE configuration setting, [459](#)
 import (MaxL), [636](#)
 IN. *See* Operators (MDX)
 INCEMPTYROWS report writer command, [1262](#)
 INCFORMATS report writer command, [1262](#)
 INCMASK report writer command, [1263](#)
 INCMISSINGROWS report writer command, [1263](#)
 incremental data loads, [752](#)
 INCRESTRUC configuration setting, [460](#)
 INCZEROROWS report writer command, [1263](#)
 INDENT report writer command, [1264](#)
 INDENTGEN report writer command, [1265](#)
 INDEXCACHESIZE configuration setting, [463](#)
 Int (integer) function

- in calculation scripts or formulas, [100](#)
- in MDX queries, [1071](#)

 intelligent calculation

- clean/dirty status, [341](#)
- turning on/off, [366](#)

 Interest calculation function, [101](#)
 Intersect MDX function, [1072](#)
 IPARENT report writer command, [1267](#)
 Irdescendants calculation function, [102](#)
 IRR calculation function, [103](#)
 Irsiblings calculation function, [104](#)
 IS. *See* Operators (MDX)
 IS MDX function, [1074](#)
 Isacctype function

- in calculation scripts or formulas, [105](#)
- in MDX queries, [1075](#)

 Isancest calculation function, [106](#)
 Isancestor MDX function, [1075](#)
 Isattribute function

- in calculation scripts or formulas, [106](#)

 Ischild function

- in calculation scripts or formulas, [107](#)
- in MDX queries, [1076](#)

Isdesc calculation function, [107](#)
 IsEmpty MDX function, [1077](#)
 Isgen calculation function, [108](#)
 Isgeneration MDX function, [1078](#)
 Isiancest calculation function, [108](#)
 isiblings calculation function, [109](#)
 Isichild calculation function, [110](#)
 Isidesc calculation function, [110](#)
 Isiparent calculation function, [111](#)
 Isisibling calculation function, [111](#)
 IsLeaf MDX function, [1079](#)
 Islev calculation function, [112](#)
 Islevel MDX function, [1079](#)
 IsMatch MDX function, [1080](#)
 Ismbr calculation function, [112](#)
 Ismbrwithattr function
 in calculation scripts or formulas, [113](#)
 Isparent calculation function, [114](#)
 Issamegen calculation function, [115](#)
 Issamelev calculation function, [115](#)
 Issibling function
 in calculation scripts or formulas, [116](#)
 in MDX queries, [1082](#)
 Isuda function
 in calculation scripts or formulas, [117](#)
 in MDX queries, [1083](#)
 IsValid MDX function, [1083](#)
 Item MDX function, [1084](#)
 iterative calculations, [335](#)

J

JulianDate
 MDX function, [1086](#)
 JVMODULELOCATION configuration setting,
[463](#)

K

keywords in MaxL, [767](#)

L

Lag MDX function, [1087](#)
 lancestors calculation function, [117](#)
 last members, calculating, [313](#)
 LastChild MDX function, [1089](#)
 LastPeriods MDX function, [1090](#)
 LastSibling MDX function, [1091](#)

LATEST report writer command, [1267](#)
 layers in MDX
 defined, [959](#)
 functions returning a layer, [989](#)
 ldescendants calculation function, [119](#)
 Lead MDX function, [1091](#)
 leading &.. *See* ampersand
 leaf members
 MDX function testing for, [1079](#)
 Lev calculation function, [121](#)
 LEV report writer command, [1269](#)
 Level MDX function, [1097](#)
 levels
 calculation functions pertaining to, [72](#), [112](#), [115](#),
 [121](#)
 described, [21](#)
 in MaxL, [959](#)
 MDX functions pertaining to, [1079](#), [1097](#), [1098](#)
 Levels MDX function, [1098](#)
 Levmbros calculation function, [121](#)
 like calculation function, [123](#)
 limits
 in MDX queries, [971](#)
 LINK report writer command, [1270](#)
 linked reporting object (LRO) operations in MaxL,
[640](#)
 LinkMember function
 in MDX queries, [1099](#)
 List calculation function, [124](#)
 LMARGIN report writer command, [1272](#)
 Ln function
 in calculation scripts or formulas, [125](#)
 in MDX queries, [1100](#)
 loading data with MaxL, [752](#)
 loading dimensions with MaxL, [754](#)
 local currency, converting, [314](#)
 locale headers, [1377](#)
 location aliases
 creating in MaxL, [688](#)
 deleting in MaxL, [733](#)
 displaying in MaxL, [715](#)
 locking files on UNIX, [446](#)
 locks
 deleting in MaxL, [733](#)
 displaying in MaxL, [715](#)
 limiting number of blocks that can be locked, [401](#)
 limiting the time users can hold, [464](#)

LOCKTIMEOUT configuration setting, [464](#)

log files

logging query durations, [507](#)

login failure details in, [465](#)

MaxL Shell log files, [833](#)

of calculation progress, [363](#), [403](#)

of outline changes, [478](#), [479](#)

of outline queries, [497](#)

of spreadsheet updates, [499](#), [500](#)

to overwrite or not, [410](#)

Log function

in calculation scripts or formulas, [125](#)

in MDX queries, [1101](#)

Log10 function

in calculation scripts or formulas, [126](#)

in MDX queries, [1101](#)

logging in

error detail upon failure, [465](#)

from ESSCMD, [526](#)

from MaxL Shell, [819](#)

LOGINFAILUREMESSAGEDETAILED

configuration setting, [465](#)

LOGMESSAGELEVEL configuration setting, [465](#)

LOOP...ENDLOOP calculation command, [335](#)

LROONSHAREDMBR configuration setting, [466](#)

lsiblings calculation function, [127](#)

M

MACRO-NAME in MaxL, [767](#)

macros (custom defined)

creating in MaxL, [689](#)

deleting in MaxL, [734](#)

displaying in MaxL, [716](#)

MASK report writer command, [1273](#)

Match calculation function, [128](#)

MATCH report writer command, [1275](#)

MATCHEX report writer command, [1276](#)

materializing aggregations, [738](#)

mathematical functions

in calculator, [27](#)

in MDX, [989](#)

Max (maximum value) function

in calculation scripts or formulas, [129](#)

in MDX queries, [1102](#)

MAX_REQUEST_GRID_SIZE configuration setting, [468](#)

MAX_RESPONSE_GRID_SIZE configuration setting, [469](#)

MAXERRORMBRVERIFYREPORT configuration setting, [467](#)

MAXFORMULACACHESIZE configuration setting, [467](#)

maximums

calculation functions, [129](#), [130](#), [131](#), [154](#)

Max MDX function, [1102](#)

MaxL

help with diagrams, [624](#)

migrating from ESSCMD, [848](#)

overview, [626](#)

starting, [819](#)

statements, [626](#)

MaxL data definition language

help with diagrams, [624](#)

migrating from ESSCMD, [848](#)

overview, [626](#)

starting, [819](#)

statements, [626](#)

MaxL Perl Module, [841](#)

MaxL Shell

commands, [833](#)

invoking, [819](#)

overview, [819](#)

syntax rules, [828](#)

MAXLOGINS configuration setting, [468](#)

Maxrange calculation function, [129](#)

Maxs calculation function, [130](#)

Maxsrange calculation function, [131](#)

mbrcompare calculation function, [133](#)

mbrparent calculation function, [135](#)

Mdallocate calculation function, [136](#)

Mdancestval calculation function, [140](#)

Mdparentval calculation function, [141](#)

Mdshift calculation function, [142](#)

MDX

comments, [970](#)

functions, [989](#)

outline formulas, [973](#)

overview, [931](#)

properties, [964](#)

query format, [932](#)

syntax, [934](#)

MDXFORMULARECURSIONLIMIT configuration setting, [470](#)

- MEANINGLESSTEXT report writer command, 1278
 - Median calculation function, 143
 - Member calculation function, 145
 - member formula calculation commands, 305
 - member formulas.. *See* formulas
 - member set functions, 27
 - MEMBER-NAME in MaxL, 767
 - MemberRange MDX function, 1104
 - members
 - formulas in calculation scripts, 305
 - names/aliases in reports, 1171
 - selecting/sorting in reports, 1171
 - specifying in MDX queries, 960
 - Members MDX function, 1106
 - MEMSCALINGFACTOR configuration setting, 471
 - Merge calculation function, 145
 - merging data slices (aggregate storage), 873
 - messaging
 - for calculations, 360
 - metadata filtering, 926
 - metadata security, 926
 - metaread access in filters, 684, 926
 - metaread permission, 926
 - Min (minimum value) function
 - in calculation scripts or formulas, 147
 - in MDX queries, 1107
 - minimums
 - calculation functions, 147, 148, 149, 158
 - Min MDX function, 1107
 - mining data, 907
 - Minrange calculation function, 147
 - Mins calculation function, 148
 - Minsrange calculation function, 149
 - missing values
 - aggregating, 336
 - clearing blocks, 316
 - clearing data, 317
 - determining whether a value is empty in MDX queries, 1077
 - displaying empty rows in Report Writer, 1262
 - displaying missing rows in Report Writer, 1263
 - substituting with a text label in Report Writer, 1278
 - suppressing in MDX queries, 949
 - missing values (#MISSING)
 - aggregating, 336
 - clearing blocks, 316
 - clearing data, 317
 - displaying empty rows in reports, 1262
 - displaying missing rows in reports, 1263
 - how they are treated in calculation scripts or formulas, 25
 - setting calculated row name as, 1237
 - substituting with text label in reports, 1278
 - MISSINGTEXT report writer command, 1278
 - Mod (modulus) function
 - in calculation scripts or formulas, 151
 - in MDX queries, 1108
 - Mode calculation function, 151
 - models for data mining
 - creating, 911
 - deleting, 919
 - displaying, 916
 - monitoring calculation progress, 360, 363
 - monitoring data, 699, 927
 - Movavg calculation function, 153
 - Movmax calculation function, 154
 - Movmed calculation function, 156
 - Movmin calculation function, 158
 - Movsum calculation function, 159
 - Movsumx calculation function, 161
 - MULTIPLEBITMAPMEMCHECK configuration setting, 472
- ## N
- Name calculation function, 163
 - named sets in MDX, 954
 - NAMESCOL report writer command, 1279
 - NAMESON report writer command, 1280
 - NAMEWIDTH report writer command, 1280
 - nested IF statements, 333
 - Netbinretrydelay configuration setting, 473
 - NETDELAY configuration setting, 473
 - NETRETRYCOUNT configuration setting, 474
 - NETSSLHANDSHAKETIMEOUT configuration setting, 474
 - NETTCPCONNECTRETRYCOUNT configuration setting, 475
 - NEWPAGE report writer command, 1281
 - Next calculation function, 164
 - NextMember MDX function, 1109
 - Nexths calculation function, 165
 - nextsibling calculation function, 166
 - NOINDENTGEN report writer command, 1282

NOMSGLOGGINGONDATAERRORLIMIT
 configuration setting, [476](#)
 NonEmptyCount MDX function, [1111](#)
 NOPAGEONDIMENSION report writer command,
[1282](#)
 NOROWREPEAT report writer command, [1283](#)
 NOSKIPONDIMENSION report writer command,
[1284](#)
 NOT, [24](#), [963](#)
 notequal calculation function, [167](#)
 NOUNAMEONDIM report writer command, [1285](#)
 NPV calculation function, [168](#)
 NUMBEROFSECFILEBACKUPS configuration
 setting, [476](#)
 numbers in MaxL, [767](#)
 numeric value expressions as arguments to MDX
 functions, [998](#), [1004](#), [1006](#), [1007](#), [1009](#), [1010](#), [1018](#),
[1045](#), [1064](#), [1071](#), [1090](#), [1100](#), [1101](#), [1102](#), [1107](#),
[1108](#), [1111](#), [1116](#), [1123](#), [1128](#), [1130](#), [1138](#), [1147](#),
[1148](#), [1149](#), [1150](#)
 numeric value functions in MDX, [989](#)
 NUMERICPRECISION configuration setting, [477](#)
 NumToStr MDX function, [1115](#)

O

OBJECT-NAME in MaxL, [767](#)
 objects (database files)
 deleting in MaxL, [734](#)
 displaying in MaxL, [717](#)
 modifying in MaxL, [663](#)
 OFFCOLCALCS report writer command, [1285](#)
 OFFROWCALCS report writer command, [1286](#)
 OFSAMEGEN report writer command, [1286](#)
 ON COLUMNS/ON ROWS/ON PAGES etc, [949](#)
 ONCOLCALCS report writer command, [1287](#)
 ONROWCALCS report writer command, [1288](#)
 ONSAMELEVELAS report writer command, [1288](#)
 OpeningPeriod MDX function, [1115](#)
 operators
 calculation, [303](#)
 in MDX, [963](#)
 OR, [24](#), [963](#)
 Order MDX function, [1116](#)
 ORDER report writer command, [1289](#)
 ORDERBY report writer command, [1291](#)
 Ordinal MDX function, [1117](#)
 OUTALT report writer command, [1293](#)

OUTALTMBR report writer command, [1294](#)
 OUTALTNames report writer command, [1295](#)
 OUTALTSELECT report writer command, [1297](#)
 OUTFORMATTEDMISSING report writer
 command, [1298](#)
 OUTFORMATTEDVALUES report writer command,
[1298](#)
 outline
 change log file size, [479](#)
 conversion to aggregate storage, [884](#)
 log of changes, [478](#)
 log of queries, [497](#)
 MDX formulas for aggregate storage outlines, [973](#)
 synchronization of partitioned outlines, [764](#)
 unlocking, [663](#)
 verification errors, [467](#)
 OUTLINECHANGELOG configuration setting, [478](#)
 OUTLINECHANGELOGFILESIZE configuration
 setting, [479](#)
 OUTMBRALT report writer command, [1298](#)
 OUTMBRNames report writer command, [1300](#)
 OUTMEANINGLESS report writer command, [1300](#)
 output
 generating report output, [1300](#)
 logging output of MaxL Shell, [833](#)
 OUTPUT command
 ESSCMD, [590](#)
 report writer, [1300](#)
 OUTPUTMEMBERKEY report command, [1301](#)

P

page breaks in reports
 inserting, [1281](#)
 when PAgELength is exceeded, [1251](#), [1346](#)
 page headings in reports
 defining in place of current, [1338](#)
 displaying, [1256](#)
 displaying before next row, [1302](#)
 displaying immediately, [1261](#)
 ending definition of, [1249](#)
 suppressing, [1346](#)
 suppressing with other formatting, [1341](#)
 PAGE report writer command, [1301](#)
 PAGEHEADING report writer command, [1302](#)
 PAgELength report writer command, [1304](#)
 PAGEONDIMENSION report writer command,
[1304](#)

- pages in report writer
 - column headings, [1239](#)
 - inserting, [1281](#)
 - row formatting, [1319](#)
 - setting center lines, [1325](#)
 - setting max number of lines, [1304](#)
 - parallel calculation, [338](#)
 - ParallelPeriod MDX function, [1119](#)
 - PARCALCMULTIPLEBITMAPMEMCHECK configuration setting, [479](#)
 - Parent function
 - in calculation scripts or formulas, [169](#)
 - in MDX queries, [1119](#)
 - PARENT report writer command, [1306](#)
 - parents
 - adding to reports, [1306](#)
 - Parentval calculation function, [170](#)
 - partial calculations, [331](#)
 - partitions
 - creating in MaxL, [691](#)
 - deleting in MaxL, [735](#)
 - displaying in MaxL, [718](#)
 - modifying in MaxL, [664](#)
 - using port numbers in host names, [923](#)
 - PASSWORD in MaxL, [767](#)
 - percent functions in MDX queries, [1007](#), [1148](#)
 - performance statistics
 - output (ESSCMD), [572](#)
 - output (MaxL), [758](#)
 - settings (ESSCMD), [603](#)
 - settings (MaxL), [649](#)
 - period-to-date calculation function, [174](#)
 - periods (in MDX queries), [1016](#), [1115](#), [1119](#), [1121](#), [1159](#)
 - Periodstodate MDX function, [1121](#)
 - Perl module, [841](#)
 - PERSISTUSERATLOGIN configuration setting, [480](#)
 - PERSPECTIVE report writer command, [1306](#)
 - PIPEBUFFERSIZE configuration setting, [480](#)
 - port numbers
 - in host names, [767](#)
 - in partition host names, [923](#)
 - PORTINC configuration setting, [481](#)
 - PORTUSAGELOGINTERVAL configuration setting, [482](#)
 - Power function
 - in calculation scripts or formulas, [171](#)
 - in MDX queries, [1123](#)
 - Preloadaliasesnamespace configuration setting, [483](#)
 - Preloadmembertnamespace configuration setting, [483](#)
 - PRELOADUDANAMESPACE configuration setting, [484](#)
 - PrevMember MDX function, [1123](#)
 - prevsibling calculation function, [171](#)
 - PRINTROW report writer command, [1307](#)
 - Prior calculation function, [172](#)
 - Priors calculation function, [173](#)
 - privileges
 - defined, [767](#)
 - displaying in MaxL, [719](#)
 - granting in MaxL, [750](#)
 - PTD function
 - in calculation scripts or formulas, [174](#)
 - in MDX queries, [1159](#)
 - PYRAMIDHEADERS report writer command, [1308](#)
- ## Q
- QRYGOVEXECBLK configuration setting, [485](#)
 - QRYGOVEXECTIME configuration setting, [486](#)
 - queries
 - against relational databases.. *See* Hybrid Analysis.
 - for data and metadata using MDX, [931](#)
 - for database state using MaxL, [758](#)
 - timing, [507](#)
 - tracking, [1161](#)
 - outline queries, [497](#)
 - query (MaxL), [636](#)
 - query limits
 - in MDX, [971](#)
 - quotation marks
 - used with ESSCMD commands, [521](#)
 - QUOTEMBRNAMES report writer command, [1309](#)
- ## R
- railroad diagrams in MDX documentation, [624](#)
 - Range calculation function, [175](#)
 - range functions, [27](#)
 - Rank calculation function, [177](#)
 - Rdescendants calculation function, [178](#)
 - RealValue function
 - in MDX queries, [1126](#)
 - refresh (MaxL), [636](#)

- reiterating commands, [335](#)
 - relationship functions
 - in calculator, [27](#)
 - in MDX queries, [1000](#), [1015](#), [1020](#), [1032](#), [1053](#), [1075](#), [1076](#), [1082](#), [1089](#), [1119](#), [1130](#)
 - Relative calculation function, [179](#)
 - RelMemberRange MDX function, [1127](#)
 - Remainder function
 - in calculation scripts or formulas, [180](#)
 - in MDX queries, [1128](#)
 - Remove calculation function, [181](#)
 - REMOVECOLCALCS report writer command, [1310](#)
 - RENAME report writer command, [1310](#)
 - REPALIAS report writer command, [1311](#)
 - REPALIASMBR report writer command, [1312](#)
 - repeating commands, [335](#)
 - REPLAYSECURITYOPTION configuration setting, [488](#)
 - REPLICATIONASSUMEIDENTICALOUTLINE configuration setting, [489](#)
 - REPMBR report writer command, [1313](#)
 - REPMBRALIAS report writer command, [1315](#)
 - report layout commands, [1171](#)
 - report scripts
 - delimiters in, [1170](#)
 - running in MaxL, [743](#)
 - syntax, [1170](#)
 - Report Writer
 - command groups, [1171](#)
 - overview, [1169](#)
 - syntax, [1170](#)
 - report writer command groups, [1171](#)
 - reports
 - alias and member name display, [1171](#)
 - column or row calculation, [1171](#)
 - data ordering, [1171](#)
 - data range, [1171](#)
 - delimiters, [1170](#)
 - format, [1171](#)
 - layout, [1171](#)
 - member selection/sorting, [1171](#)
 - output, [1216](#)
 - syntax, [1170](#)
 - REPQUALMBR report writer command, [1316](#)
 - RESTRICT report writer command, [1317](#)
 - results for data mining
 - deleting, [918](#)
 - displaying, [915](#)
 - retrieval buffer
 - dynamic sizing, [514](#)
 - Return calculation function, [182](#)
 - Round function
 - in calculation scripts or formulas, [183](#)
 - in MDX queries, [1130](#)
 - row calculation commands, [1171](#)
 - ROW report writer command, [1319](#)
 - ROWREPEAT report writer command, [1320](#)
 - rsiblings calculation function, [184](#)
 - RTDEPCALCOPTIMIZE configuration setting, [490](#)
- ## S
- Sancestval calculation function, [185](#)
 - SAVEANDOUTPUT report writer command, [1321](#)
 - SAVEROW report writer command, [1323](#)
 - SCALE report writer command, [1324](#)
 - SECFILEBACKUPINTERVAL configuration setting, [490](#)
 - security file compaction
 - triggering by defragmentation percent, [491](#)
 - triggering using MaxL, [668](#)
 - security file export (MaxL), [749](#)
 - security privileges
 - defined, [767](#)
 - displaying in MaxL, [719](#)
 - granting in MaxL, [750](#)
 - Securityfilecompactionpercent configuration setting, [491](#)
 - SELECT statements (MDX), [932](#)
 - semicolon
 - used with ESSCMD commands, [521](#)
 - serial calculation, [338](#)
 - SERVERLEASEEXPIRATIONTIME configuration setting, [492](#)
 - SERVERLEASEMAXRETRYCOUNT configuration setting, [492](#)
 - SERVERLEASERENEWALTIME configuration setting, [493](#)
 - SERVERPORTBEGIN configuration setting, [493](#)
 - SERVERPORTEND configuration setting, [494](#)
 - SERVERTHREADS configuration setting, [496](#)
 - SESSION-ID in MaxL, [767](#)
 - sessions
 - displaying in MaxL, [720](#)
 - stopping in MaxL, [668](#)

- sessions for data mining
 - displaying, 916
 - initiating, 911
 - stopping, 668
- SET AGGMISG calculation command, 336
- SET CACHE calculation command, 337
- SET CALCPARALLEL calculation command, 338
- SET CALCTASKDIMS calculation command, 339
- SET CCTRACKCALC calculation command, 341
- SET CLEARUPDATESTATUS calculation command, 341
- SET commands (overview), 336
- SET COPYMISSINGBLOCK calculation command, 344
- SET CREATEBLOCKONEQ calculation command, 346
- SET CREATENONMISSINGBLK calculation command, 345
- SET DATAEXPORTOPTIONS calculation command, 348
- SET DATAIMPORTIGNORETIMESTAMP calculation command, 356
- SET EMPTYMEMBERSETS calculation command, 357
- SET FRMLBOTTOMUP calculation command, 358
- SET FRMLRTDYNAMIC calculation command, 358
- set functions in MDX, 989
- SET LOCKBLOCK calculation command, 359
- SET MSG calculation command, 360
- SET NOTICE calculation command, 363
- SET REMOTECALC calculation command, 365
- SET SCAPERSPECTIVE calculation command, 365
- SET UPDATECALC calculation command, 366
- SET UPTOLOCAL calculation command, 367
- SETCENTER report writer command, 1325
- SETROWOP report writer command, 1325
- sets in MDX, 953
 - limits, 971
- Share calculation function, 187
- shared members
 - returning, 1099
- Shift calculation function, 187
- Shiftminus calculation function, 188
- Shiftplus calculation function, 189
- shiftsibling calculation function, 190
- Siblings function
 - in MDX queries, 1130
- siblings function
 - in calculation scripts or formulas, 191
- Silentotlquery configuration setting, 497
- SINGLECOLUMN report writer command, 1327
- SIZE-STRING in MaxL, 767
- SKIP report writer command, 1327
- SKIPONDIMENSION report writer command, 1328
- slicers in MDX, 952
- SLN calculation function, 192
- solve order in MDX queries, 954
- SORTALTNAMES report writer command, 1329
- SORTASC report writer command, 1331
- SORTDESC report writer command, 1331
- SORTGEN report writer command, 1333
- SORTLEVEL report writer command, 1334
- SORTMBRNames report writer command, 1336
- SORTNONE report writer command, 1336
- Sparentval calculation function, 193
- sparse dimensions, calculating
 - calculating only, 309
 - parallel calculation and, 339
- sparse formula calculation, 358
- SPARSE report writer command, 1337
- special characters in MaxL, 767
- Spline calculation function, 195
- SPLITARCHIVEFILE configuration setting, 497
- spreadsheet
 - error messages, 503
 - logging queries, 1161
 - logging updates, 499, 500
 - optimized grid processing, 503
 - processing controls, 504
 - suppressing grid messages, 448
 - timing queries, 507
- SQLFETCHERRORPOPUP configuration setting, 498
- SSAUDIT configuration setting, 499
- SSAUDITR configuration setting, 500
- SSINVALIDTEXTDETECTION configuration setting, 501
- SSL configuration settings
 - AGENTSECUREREPORT, 388
 - CLIENTPREFERREDMODE, 411
 - ENABLECLEARMODE, 439
 - ENABLESECUREMODE, 440
 - NETSSLHANDSHAKETIMEOUT, 474
 - SSLCIPHERSUITES, 502

WALLET_PATH, 515
 SSLCIPHERSUITES configuration setting, 502
 SSLOGUNKNOWN configuration setting, 503
 SSOPTIMIZEDGRIDPROCESSING config setting, 503
 SSPROCROWLIMIT configuration setting, 504
 STARTHEADING report writer command, 1338
 statements in MaxL, 626
 statistical calc functions, 27
 status of clean/dirty blocks, 341
 Stddev MDX function, 1132
 StddevP MDX function, 1133
 Stdev calculation function, 199
 Stdevp calculation function, 200
 Stdevrange calculation function, 201
 Subset MDX function, 1136
 substitution variables
 adding in MaxL, 642, 653, 668
 deleting in MaxL, 642, 653, 668
 displaying in MaxL, 729
 setting in MaxL, 642, 649, 653, 668
 using "&" character with, 309, 1216
 Substring calculation function, 203
 SUDA report writer command, 1340
 Sum function
 in calculation scripts or formulas, 203
 in MDX queries, 1138
 sum functions in MDX queries, 1009, 1138, 1149
 Sumrange calculation function, 204
 SUPALL report writer command, 1341
 SUPBRACKETS report writer command, 1343
 SUPCOLHEADING report writer command, 1343
 SUPCOMMAS report writer command, 1344
 SUPCURHEADING report writer command, 1344
 SUPEMPTYROWS report writer command, 1345
 SUPEUROPEAN report writer command, 1345
 SUPFEED report writer command, 1346
 SUPFORMATS report writer command, 1346
 SUPHEADING report writer command, 1346
 SUPMASK report writer command, 1347
 SUPMISSINGROWS report writer command, 1347
 SUPNA configuration setting, 505
 SUPNAMES report writer command, 1348
 SUPOUTPUT report writer command, 1349
 SUPPAGEHEADING report writer command, 1350
 SUPSHARE report writer command, 1351
 SUPSHAREOFF report writer command, 1352

SUPZEROROWS report writer command, 1353
 SYD calculation function, 205
 SYM report writer command, 1354
 syntax
 report script, 1170

T

TABDELIMIT report writer command, 1355
 tablespaces
 altering, 675
 displaying, 885
 tabs in reports, 1355
 Tail MDX function, 1139
 TARGETASOOPT configuration setting, 506
 TARGETTIMESERIESOPT configuration setting, 507
 templates for data mining
 creating, 911
 deleting, 919
 displaying, 916
 temporary variables in calculations
 containing a single value, 368
 one-dimensional array, 310
 terminals in MaxL, 767
 TEXT report writer command, 1356
 threads
 parallel calculation and, 338
 time balance members, calculating
 average, 311
 first, 313
 in MDX queries, 1016, 1090, 1115, 1119, 1121, 1159
 last, 313
 TIMINGMESSAGES configuration setting, 507
 Todate function
 in calculation scripts or formulas, 206
 in MDX queries, 1142
 TODATE report writer command, 1361
 TOP report writer command, 1361
 TopCount MDX function, 1147
 TopPercent MDX function, 1148
 TopSum MDX function, 1149
 TRANSACTIONLOGDATALOADARCHIVE configuration setting, 508
 TRANSACTIONLOGLOCATION configuration setting, 510
 Trend calculation function, 210

TRIGGER-NAME in MaxL, [767](#)

triggers

about, [927](#)

creating in MaxL, [699](#)

deleting in MaxL, [736](#)

displaying in MaxL, [724](#)

modifying in MaxL, [676](#)

TRIGMAXMEMSIZE configuration setting, [511](#)

Truncate function

in calculation scripts or formulas, [220](#)

in MDX queries, [1150](#)

TupleRange MDX function, [1150](#)

tuples in MDX, [961](#)

two-pass calculation, [314](#)

U

UCHARACTERS report writer command, [1363](#)

UCOLUMNS report writer command, [1364](#)

UDA function

in calculation scripts or formulas, [220](#)

in MDX queries, [1151](#)

UDA report writer command, [1365](#)

UDAs

using in calculation scripts or formulas, [117](#), [220](#)

using in MDX queries, [964](#), [1083](#), [1151](#)

UDATA report writer command, [1366](#)

UNAME report writer command, [1367](#)

UNAMEONDIMENSION report writer command, [1368](#)

UNDERLINECHAR report writer command, [1369](#)

underlining in reports

changed row member names in same dimension, [1368](#)

characters in preceding row, [1363](#)

data columns only, [1366](#)

default, [1369](#)

name and data columns, [1364](#)

name columns only, [1367](#)

turning off for new member names in same dimension, [1285](#)

UNDERSCORECHAR report writer command, [1370](#)

Unicode, [1377](#)

UNICODEAGENTLOG configuration setting, [512](#)

Union MDX function, [1153](#)

UNIX

locking files on, [446](#)

UnixDate

MDX function, [1154](#)

UPDATECALC configuration setting, [513](#)

user sessions

displaying in MaxL, [720](#)

stopping in MaxL, [668](#)

USER-NAME in MaxL, [767](#)

users

creating in MaxL, [702](#)

deleting in MaxL, [736](#)

displaying in MaxL, [725](#)

modifying in MaxL, [676](#)

UTF-8 conversion utility, [1377](#)

V

Value MDX function, [1155](#)

value without attribute association

MDX functions, [1126](#)

VAR calculation command, [368](#)

VARIABLE-NAME in MaxL, [767](#)

variables

array variables, [310](#)

temporary in calculations, [368](#)

variance function

in calculation scripts or formulas

statistical variance, [222](#)

variance as "difference", [221](#)

Variancep calculation function, [223](#)

Varper calculation function, [221](#)

varying attributes

calculation function that returns base members with varying attributes, [106](#)

calculation function that returns members with type of attribute, [225](#)

calculation function that returns members with type of varying attribute, [113](#)

defining perspective in calculation scripts, [365](#)

MDX function that returns members with attribute based on perspective, [1003](#)

MDX function that returns members with type of varying attribute, [1158](#)

Vlbreport configuration setting, [514](#)

VOLUME-NAME in MaxL, [767](#)

W

WALLETPATH configuration setting, [515](#)

WIDTH report writer command, [1370](#)

wild card search

MDX functions pertaining to, [1080](#)

WITH section in MDX queries, [954](#)

Withattr function

in calculation scripts or formulas, [225](#)

in MDX queries, [1157](#)

WITHATTR report writer command, [1372](#)

WithattrEx function

in MDX queries, [1158](#)

WITHATTREX report writer command, [1373](#)

X

XML configuration

for query logging, [1162](#)

XOLAPMAXNUMCONNECTION configuration

setting, [516](#)

XOLAPSCHEMAVERIFICATION configuration

setting, [516](#), [517](#)

XOLAPSQLIDLEPERIOD configuration setting, [518](#)

Xrange calculation function, [227](#)

Xref calculation function, [229](#)

Xwrite calculation function, [232](#)

Z

zero values in reports, [1374](#)

ZEROTEXT report writer command, [1374](#)