

Oracle® Essbase

Scripting Reference for Oracle Essbase



F17647-17
December 2024



Oracle Essbase Scripting Reference for Oracle Essbase,

F17647-17

Copyright © 2019, 2024, Oracle and/or its affiliates.

Primary Author: Essbase 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 is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware 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 that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about 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 unless otherwise set forth in an applicable agreement between you and Oracle. 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, except as set forth in an applicable agreement between you and Oracle.

Contents

1 Scripting Reference Overview

About the Scripting Reference	1-1
About Aggregate Storage Cubes	1-1

2 Essbase Command-Line Interface (CLI)

Download and Use the Command-Line Interface	2-1
CLI Command Reference	2-2
Login/Logout: CLI Authentication	2-3
Calc: Run a Calculation Script	2-4
Clear: Remove Data from a Cube	2-5
Createlocalconnection: Save a JDBC Connection	2-6
Dataload: Load Data to a Cube	2-8
Deletefile: Remove Cube Files	2-9
Deploy: Create a Cube from a Workbook	2-10
Dimbuild: Load Dimensions to a Cube	2-11
Download: Get Cube Files	2-13
Help: Display Command Syntax	2-14
LcmExport: Back Up Cube Files	2-14
LcmImport: Restore Cube Files	2-16
Listapp: Display Applications	2-18
Listdb: Display Cubes	2-18
Listfiles: Display Files	2-18
Listfilters: View Security Filters	2-19
Listlocks: View Locks	2-20
Listvariables: Display Substitution Variables	2-20
Setpassword: Store CLI Credentials	2-21
Start: Start an Application or Cube	2-21
Stop: Stop an Application or Cube	2-22
Unsetpassword: Remove Stored CLI Credentials	2-22
Upload: Add Cube Files	2-22
Version: Display API Version	2-24

3 MaxL

Manage Essbase Using the MaxL Client	3-1
Prerequisites to Set Up the MaxL Client	3-1
Download and Use the MaxL Client	3-3
How to Read MaxL Railroad Diagrams	3-4
Anatomy of MaxL Statements	3-5
Railroad Diagram Symbols	3-5
Sample Railroad Diagram	3-6
MaxL Statements	3-7
Listed By Verbs	3-7
Alter	3-8
Create	3-8
Display	3-9
Drop	3-9
Execute	3-10
Export	3-10
Grant	3-10
Import	3-10
Login	3-11
Query	3-11
Refresh	3-12
Listed by Objects	3-12
Aggregate Build	3-13
Aggregate Process	3-13
Aggregate Selection	3-13
Allocation	3-13
Application	3-13
Archive_file	3-14
Calculation	3-14
Custom Definitions	3-14
Data	3-14
Database	3-14
Dimensions	3-15
Drillthrough	3-15
Filter	3-15
Function	3-15
Group	3-15
Location Alias	3-16
Lock	3-16
LRO	3-16
Macro	3-16

Object	3-16
Outline	3-17
Partition	3-17
Privilege	3-17
Query_tracking	3-17
Session	3-17
System	3-18
Tablespace	3-18
Trigger	3-18
Trigger Spool	3-18
User	3-18
Variable	3-18
MaxL Statement Reference	3-19
Alter Application	3-19
Alter Database	3-24
Alter Database enable disable	3-25
Alter Database Set	3-28
Alter Database (Misc)	3-33
Alter Drillthrough	3-39
Alter Filter	3-40
Alter Group	3-41
Alter Object	3-42
Alter Partition	3-43
Alter Session	3-46
Alter System	3-49
Alter Trigger	3-57
Alter User	3-57
Create Application	3-58
Create Calculation	3-59
Create Database	3-61
Create Drillthrough	3-62
Create Filter	3-63
Create Function	3-65
Create Group	3-67
Create Location Alias	3-67
Create Macro	3-69
Create Partition	3-70
Create Replicated Partition	3-71
Create Transparent Partition	3-74
Create Trigger	3-77
Create After-Update Trigger	3-77
Create On-Update Trigger	3-78

Create User	3-80
Display Application	3-81
Display Calculation	3-83
Display Database	3-84
Display Drillthrough	3-89
Display Filter	3-90
Display Filter Row	3-91
Display Function	3-92
Display Location Alias	3-94
Display Lock	3-95
Display Macro	3-96
Display Object	3-98
Display Partition	3-100
Display Privilege	3-100
Display Session	3-102
Display System	3-104
Display Trigger	3-108
Display Trigger Spool	3-109
Display Variable	3-110
Drop Application	3-111
Drop Calculation	3-112
Drop Database	3-113
Drop Drillthrough	3-113
Drop Filter	3-114
Drop Function	3-114
Drop Group	3-115
Drop Location Alias	3-115
Drop Lock	3-116
Drop Macro	3-118
Drop Object	3-118
Drop Partition	3-119
Drop Trigger	3-120
Drop Trigger Spool	3-121
Drop User	3-121
Execute Calculation	3-122
Export Data	3-124
Export LRO	3-127
Export Outline	3-129
Grant	3-132
Import Data	3-135
Import Dimensions	3-138
Import LRO	3-141

Query Application	3-142
Query Archive_File	3-143
Query Database	3-143
Refresh Custom Definitions	3-149
Refresh Outline	3-150
Refresh Replicated Partition	3-152
Performance Statistics in MaxL	3-153
MaxL Statements (Aggregate Storage)	3-158
Alter Application (Aggregate Storage)	3-160
Alter Database (Aggregate Storage)	3-165
Alter System (Aggregate Storage)	3-174
Alter Tablespace (Aggregate Storage)	3-181
Create Application (Aggregate Storage)	3-183
Create Database (Aggregate Storage)	3-185
Create Outline (Aggregate Storage)	3-186
Display Tablespace (Aggregate Storage)	3-188
Execute Aggregate Build (Aggregate Storage)	3-188
Execute Aggregate Process (Aggregate Storage)	3-190
Execute Aggregate Selection (Aggregate Storage)	3-192
Execute Allocation (Aggregate Storage)	3-196
Execute Calculation (Aggregate Storage)	3-200
Export Data (Aggregate Storage)	3-203
Export Query Tracking (Aggregate Storage)	3-205
Import Data (Aggregate Storage)	3-207
Import Query Tracking (Aggregate Storage)	3-212
Query Application (Aggregate Storage)	3-213
Query Database (Aggregate Storage)	3-215
Outline Paging Dimension Statistics	3-226
Aggregate Storage Runtime Statistics	3-227
Aggregate Storage Slice Information Output	3-229
Aggregate Storage Group ID Information Output	3-230
Aggregate Storage Uncommitted Transaction Information Output	3-230
MaxL Definitions	3-230
MaxL Syntax Notes	3-231
Numbers in MaxL Syntax	3-232
Terminals	3-232
ACTION	3-234
ALLOC-NUMERIC	3-235
ALT-NAME-SINGLE	3-235
APP-NAME	3-236
AREA-ALIAS	3-238
BUFFER-ID	3-239

CALC-NAME	3-239
CALC-NAME-SINGLE	3-240
CALC-SPEC-STRING	3-241
CALC-STRING	3-241
COLUMN-WIDTH	3-242
COMMENT-STRING	3-242
CONDITION	3-243
CUBE-AREA or MDX-SET	3-243
DATE	3-244
DBS-EXPORT-DIR	3-245
DBS-NAME	3-245
DBS-STRING	3-247
DIM-NAME	3-248
EXPORT-DIR	3-248
FILE-NAME	3-248
FILE-NAME-PREFIX	3-249
FILTER-NAME	3-249
FULL-EXPORT-DIR	3-250
FUNC-NAME	3-251
GROUP-NAME	3-252
HOST-NAME	3-253
ID-RANGE	3-253
ID-STRING	3-254
IMP-FILE	3-254
IMPORT-DIR	3-255
JAVACLASS.METHOD	3-255
LOCATION-ALIAS-NAME	3-256
LOC-ALIAS-SINGLE	3-256
LOG-TIME	3-257
MACRO-EXPANSION	3-257
MACRO-NAME	3-257
MEMBER-EXPRESSION	3-258
MEMBER-NAME	3-259
OBJ-NAME	3-260
OBJ-NAME-SINGLE	3-260
OUTLINE-ID	3-261
PASSWORD	3-261
PATHNAME_FILENAME	3-262
PRECISION-DIGITS	3-262
PROPS	3-262
RNUM	3-263
RTSV-LIST	3-264

RULE-FILE-NAME	3-265
SESSION-ID	3-265
SIZE-STRING	3-266
SPOOL-NAME	3-266
STOPPING-VAL	3-267
TABLSP-NAME	3-267
TRIGGER-NAME	3-268
URL-NAME	3-269
USER-NAME	3-269
VARIABLE-NAME	3-270
VIEW-FILE-NAME	3-271
VIEW-ID	3-271
VIEW-SIZE	3-272
Privileges and Roles	3-272
System-Level System Privileges	3-273
System-Level System Roles	3-273
Application-Level System Roles	3-273
Database-Level System Roles	3-274
Quoting and Special Characters Rules for MaxL Language	3-275
Tokens enclosed in Single Quotation Marks	3-275
Tokens Enclosed in Double Quotation Marks	3-275
Use of Backslashes in MaxL	3-276
Use of Apostrophes (Single Quotation Marks)	3-276
Use of Dollar Signs	3-276
MaxL Shell Commands	3-276
MaxL Shell and Unicode	3-287
MaxL Shell Syntax Rules and Variables	3-287
Query Cancellation	3-291
Encryption	3-292
LoginAs	3-293
Login	3-293
ESSCMD Script Conversion	3-294
ESSCMD Script Utility Usage	3-294
Things to Note About the ESSCMD shell Script Utility	3-295
ESSCMD to MaxL Mapping	3-295
MaxL Reserved Words List	3-302
MaxL BNF	3-311
MaxL Use Cases	3-333
Create an Aggregate Storage Sample Using MaxL	3-333
Load Data Using Buffers	3-334
List Aggregate Storage Data Load Buffers	3-336
Force Deletion of Partitions	3-337

Metadata Filtering	3-338
Examples of Triggers	3-340

4 Report Writer

Report Writer Syntax	4-1
Report Delimiters	4-1
Syntax Guidelines	4-1
Referencing Static Members	4-2
Report Writer Command Groups	4-3
Report Layout Commands	4-3
Data Range Commands	4-3
Data Ordering Commands	4-4
Member Selection and Sorting Commands	4-4
Format Commands	4-5
Column or Row Calculation Commands	4-7
Member Names and Aliases	4-7
Examples of Report Scripts	4-8
Sample 1: Creating a Different Format for Each Page	4-9
Sample 2: Handling Missing Values	4-10
Sample 3: Nesting Columns	4-11
Sample 4: Grouping Rows	4-13
Sample 5: Reporting on Different Combinations of Data	4-17
Sample 6: Formatting Different Combinations of Data	4-18
Sample 7: Using Aliases	4-20
Sample 8: Creating Custom Headings and % Characters	4-22
Sample 9: Creating Custom Page Headings	4-25
Sample 10: Using Formulas	4-27
Sample 11: Placing Two-Page Layouts on the Same Page	4-28
Sample 12: Formatting for Data Export	4-30
Sample 13: Creating Asymmetric Columns	4-31
Sample 14: Calculating Columns	4-32
Sample 14-A: Basic Calculated Columns	4-32
Sample 14-B: Asymmetric Columns	4-33
Sample 15: Calculating Rows	4-34
Sample 15-A: Basic Calculated Row	4-35
Sample 15-B: Calculated Rows and Missing Relationships	4-35
Sample 15-C: Rows of Averages	4-37
Sample 16: Sorting by Top or Bottom Data Values	4-39
Sample 16-A: Bottom Data Values	4-40
Sample 16-B: Top Data Values	4-40
Sample 17: Restricting Rows	4-41

Sample 18: Ordering Data Values	4-42
Sample 19: Narrowing Member Selection Criteria	4-43
Sample 20: Using Attributes in Member Selection	4-44
Sample 21: Using the WITHATTR Command in Member Selection	4-45
Report Writer Command List	4-46
&	4-47
!	4-48
ACCOFF	4-48
ACCON	4-49
AFTER	4-50
ALLINSAMEDIM	4-51
ALLSIBLINGS	4-52
ANCESTORS	4-53
ASYM	4-54
ATTRIBUTE	4-56
ATTRIBUTEVA	4-57
BEFORE	4-58
BLOCKHEADERS	4-59
BOTTOM	4-60
BRACKETS	4-64
CALCULATE COLUMN	4-64
CALCULATE ROW	4-67
CHILDREN	4-71
CLEARALLROWCALC	4-72
CLEARROWCALC	4-73
COLHEADING	4-74
COLUMN	4-76
COMMAS	4-77
CURHEADING	4-77
CURRENCY	4-78
DATEFORMAT	4-79
DECIMAL	4-80
DESCENDANTS	4-81
DIMBOTTOM	4-83
DIMEND	4-84
DIMTOP	4-85
DUPLICATE	4-86
ENDHEADING	4-88
EUROPEAN	4-89
FEEDON	4-90
FIXCOLUMNS	4-91
FORMATCOLUMNS	4-93

GEN	4-93
HEADING	4-95
IANCESTORS	4-96
ICHILDREN	4-97
IDESCENDANTS	4-98
IMMHEADING	4-100
INCEMPTYROWS	4-101
INCFORMATS	4-102
INCMASK	4-102
INCMISSINGROWS	4-103
INCZEROROWS	4-103
INDENT	4-104
INDENTGEN	4-106
IPARENT	4-108
LATEST	4-108
LEAVES	4-109
LEV	4-111
LINK	4-112
LMARGIN	4-115
MASK	4-116
MATCH	4-118
MATCHEX	4-120
MEANINGLESSTEXT	4-121
MISSINGTEXT	4-122
NAMESCOL	4-122
NAMESON	4-124
NAMEWIDTH	4-124
NEWPAGE	4-126
NOINDENTGEN	4-126
NOPAGEONDIMENSION	4-127
NOROWREPEAT	4-128
NOSKIPONDIMENSION	4-129
NOUNAMEONDIM	4-130
OFFCOLCALCS	4-131
OFFROWCALCS	4-132
OFSAMEGEN	4-133
ONCOLCALCS	4-134
ONROWCALCS	4-135
ONSAMELEVELAS	4-135
ORDER	4-137
ORDERBY	4-138
OUTALT	4-141

OUTALTMBR	4-143
OUTALTNames	4-144
OUTALTSELECT	4-145
OUTFORMATTEDMISSING	4-147
OUTFORMATTEDVALUES	4-147
OUTMBRALT	4-148
OUTMBRNames	4-149
OUTMEANINGLESS	4-150
OUTPUT	4-150
OUTPUTMEMBERKEY	4-150
PAGE	4-151
PAGEHEADING	4-152
PAGELength	4-154
PAGEONDIMENSION	4-155
PARENT	4-157
PERSPECTIVE	4-157
PRINTROW	4-158
PYRAMIDHEADERS	4-160
QUOTEMBRNames	4-161
REMOVECOLCALCS	4-162
RENAME	4-163
REPALIAS	4-164
REPALIASMBR	4-165
REPMBR	4-167
REPMBRALIAS	4-169
REPQUALMBR	4-170
RESTRICT	4-171
ROW	4-174
ROWREPEAT	4-175
SAVEANDOUTPUT	4-176
SAVEROW	4-178
SCALE	4-180
SETCENTER	4-181
SETROWOP	4-182
SINGLECOLUMN	4-184
SKIP	4-185
SKIPONDIMENSION	4-186
SORTALTNames	4-187
SORTASC	4-189
SORTDESC	4-191
SORTGEN	4-192
SORTLEVEL	4-194

SORTMBRNames	4-196
SORTNONE	4-197
SPARSE	4-198
STARTHEADING	4-200
SUDA	4-202
SUPALL	4-203
SUPBRACKETS	4-205
SUPCOLHEADING	4-206
SUPCOMMAS	4-208
SUPCURHEADING	4-208
SUPEMPTYROWS	4-209
SUPEUROPEAN	4-210
SUPFEED	4-210
SUPFORMATS	4-211
SUPHEADING	4-211
SUPMASK	4-212
SUPMISSINGROWS	4-213
SUPNAMES	4-215
SUOUTPUT	4-216
SUPPAGEHEADING	4-217
SUPSHARE	4-218
SUPSHAREOFF	4-220
SUPZEROROWS	4-221
SYM	4-222
TABDELIMIT	4-223
TEXT	4-225
TODATE	4-228
TOP	4-229
UCHARACTERS	4-231
UCOLUMNS	4-232
UDA	4-234
UDATA	4-235
UNAME	4-236
UNAMEONDIMENSION	4-237
UNDERLINECHAR	4-239
UNDERSSCORECHAR	4-239
WIDTH	4-240
WITHATTR	4-241
WITHATTREX	4-244
ZEROTEXT	4-245
Report Writer Limits	4-246

1

Scripting Reference Overview

You can use MaxL and the Essbase Command Line Interface (CLI) to perform operations in Essbase using scripts and shell-interfaces. This reference is intended for advanced users who need detailed information and examples.

MaxL is a language interface for administering Essbase, and the CLI is a command interface. MaxL statements begin with a verb, and enable you to perform actions on Essbase artifacts. CLI commands have a variety of options to help you specify what you want to do. Report Writer is a text-based script language that you can use to report on data in cubes.

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

About the Scripting Reference

This Scripting Reference describes language and command interfaces available for Oracle Essbase. This reference is intended for advanced users who need detailed information and examples to perform operations on Essbase.

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

To use this document, you need the following:

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

About Aggregate Storage Cubes

Consider using the aggregate 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.

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 \(Aggregate Storage\)](#).

2

Essbase Command-Line Interface (CLI)

The command-line interface is a non-graphical command shell utility based on the Essbase REST API. Using the CLI, administrators can enter shell commands to perform actions on Essbase.

- [Download and Use the Command-Line Interface](#)
- [CLI Command Reference](#)

Download and Use the Command-Line Interface

Download the Command-Line Interface (CLI), available for Windows and Linux, from the desktop tools in the Console in the Essbase web interface

1. If it is not already installed, download and install Java SE Development Kit 8 from Oracle Technology Network.
2. Set the `JAVA_HOME` environment variable on your system to point to the JDK installation folder. If the installation path has any spaces, enclose the path in quotation marks. On Windows, restart the computer after setting `JAVA_HOME`.

Variable name:	JAVA_HOME
Variable value:	"C:\Program Files\Java\jdk1.8.0_321"

3. In the Essbase web interface, click **Console**.
4. In the Console, go to **Desktop Tools** and expand **Command Line Tools**.
5. Under **Command Line Tools**, click the **Command Line Interface (CLI)** tile to download the utility.
6. Save `cli.zip` to a local drive. For best results, choose a path that has no spaces; for example, `C:\Oracle`.
7. Uncompress `cli.zip`, and find the extracted files under the `cli` folder.
8. To issue commands interactively,
 - a. Navigate to the CLI folder containing the shell script, `esscs.bat` or `esscs.sh`.
 - b. If you're using a proxy, set the proxy:

For Windows:

```
set HTTPS_PROXY=www-proxy.example.com:80
```

For Linux:

```
export HTTPS_PROXY=www-proxy.example.com:80
```

c. Launch the CLI:

For Windows:

```
esscs login -u MyAdmin -p mypass7YG -url https://192.0.2.1/essbase
```

For Linux:

```
esscs.sh login -u MyAdmin -p mypass7YG -url https://192.0.2.1/essbase
```

For more examples and details, see the [login](#) command topic.

If the CLI was installed correctly, a list of supported commands is displayed.

9. To execute multiple CLI commands, add them to any shell script and execute it.

In any script you run that contains CLI commands, Oracle recommends you include the following directive before the CLI login statement:

For Windows:

```
set ESSCLI_ID=%USERNAME%_%random%
```

For Linux:

```
export ESSCLI_ID=`whoami`_$$PPID
```

This helps store session information and prevent execution errors when multiple scripts are run concurrently.

CLI Command Reference

The Essbase CLI commands that you issue in the **esscs** shell help you perform routine platform operations including: `calc`, `dataload`, `dimbuild`, `lcmexport`, `lcmimport`, `upload` and `download` of artifacts, `start` and `stop` an application or cube, and more.

The following commands are available in the command-line interface. Arguments to commands can be issued in any order.

- [calc](#)
- [clear](#)
- [createlocalconnection](#)
- [dataload](#)
- [deletefile](#)
- [deploy](#)
- [dimbuild](#)
- [download](#)
- [help](#)
- [lcmexport](#)
- [lcmimport](#)
- [listapp](#)

- `listdb`
- `listfiles`
- `listfilters`
- `listlocks`
- `listvariables`
- `login, logout`
- `setpassword`
- `start`
- `stop`
- `unsetpassword`
- `upload`
- `version`

To display help for all commands, enter `esscs -h`. To display help for a specific command, enter `esscs command -h`.

To turn on verbose output for any command, meaning that extended information (if available) is displayed, enter `esscs command -v command arguments`.

Login/Logout: CLI Authentication

The `login` CLI command for Essbase authenticates you to Essbase so you can use the CLI.

Before you can issue any other CLI commands to Essbase, you must log in. If a secure connection is required, then the URL must begin with `https`.

You can authenticate in the following ways using CLI:

- Use `setpassword` once to have the password stored for your client/user combination. In subsequent sessions, you can use the `login` command without being prompted to enter a password.
- Use the `-user` and `-password` options with the `login` command (Caution: the password appears in the shell window as cleartext).
- Use only the `-user` option with the `login` command. You are prompted to enter the password, which is hidden.

If you're a [federated](#) SSO user in Oracle Identity Cloud Service, logging in using MaxL or CLI is not supported. Federated SSO login requires a browser window. Create a native Identity Cloud Service user, and use that instead to log in using MaxL or CLI.

Syntax (`login`)

```
login [-verbose] -essbaseurl https://instance-name.example.com/essbase -user
username [-password password]
```

Option	Abbreviation	Description
<code>-verbose</code>	<code>-v</code>	Show extended descriptions
<code>-essbaseurl</code>	<code>-url</code>	Address of an instance of Essbase
<code>-user</code>	<code>-u</code>	User name

Option	Abbreviation	Description
-password	-p	Optional. Password for user. Alternatively, set the password using setpassword . If issuing the the login command from a script, and the password contains special characters, enclose it in double quotation marks (for example, "aNb3^5%9\$!"). Use of \$ (dollar sign) character within the Essbase password is not supported for logins in a Linux environment.

Example 1 (login)

```
esscs login -url https://myEssbase-test-myDomain.analytics.us2.example.com/
essbase -u smith
```

Example 2 (login)

In the following example, the user logging in, `admin1@example.com` is an Identity Cloud Service administrator who was set as the initial Essbase administrator during Essbase stack deployment on Oracle Cloud Infrastructure. As the password is not entered in this example, the administrator will be prompted to provide it next. The URL is the **essbase_url** from the job outputs resulting from the stack deployment.

```
esscs login -u admin1@example.com -url https://192.0.2.1/essbase
```

Syntax (logout)

```
logout
```

Example (logout)

```
esscs logout
```

Calc: Run a Calculation Script

The `calc` CLI command for Essbase executes a calculation script on the cube. To run this command, you need at least Database Update permission, as well as provisioned access to the calculation script.

Before you can run calculation scripts, you must first upload the scripts, as `.csc` files, to the cube directory. You can use the CLI to upload files. See [Upload: Add Cube Files](#).

Syntax

```
calc [-verbose] -application appname -db cubename -script scriptfilename
```

Option	Abbreviation	Description
-verbose	-v	Show extended descriptions
-application	-a	Application name
-db	-d	Database (cube) name

Option	Abbreviation	Description
-script	-s	Calculation script name. Must have .csc file extension. You do not need to give a full path. Files are assumed to be in the relevant cube directory.

Example

```
esscs calc -v -a Sample -d Basic -s CALCALL.CSC
```

You can also run calculation scripts using the Calculate option in Cube Designer or Smart View, Jobs in the Essbase web interface or REST API, or **execute calculation** in MaxL.

Clear: Remove Data from a Cube

The clear CLI command for Essbase clears data from a cube. To use this command, you need at least Database Update permission.

Syntax

```
clear [-verbose] -application appname -db cubename [-option clearOption[-regionspec regionSpec]]
```

Option	Abbreviation	Description
-verbose	-v	Optional. Show extended descriptions
-application	-a	Application name
-db	-d	Database (cube) name
-option	-O	Optional. Keyword specifying what to clear. Default option, if omitted, is ALL_DATA. The options for block storage cubes are: <ul style="list-style-type: none"> ALL_DATA—All data, linked objects, and the outline are cleared UPPER_LEVEL—Upper level blocks are cleared NON_INPUT—Non input blocks are cleared The options for aggregate storage cubes are: <ul style="list-style-type: none"> ALL_DATA—All data, linked objects, and the outline are cleared ALL_AGGREGATIONS —All aggregated data is cleared PARTIAL_DATA —Only specified data region is cleared. Use with -regionspec
-regionspec	-rs	MDX expression specifying the region to clear

Example

```
esscs clear -a ASOSamp -d Basic -O PARTIAL_DATA -rs "{([Jan],[Sale],[Cash])}"
```

You can also clear data using the Load Data option in Cube Designer, Jobs in the Essbase web interface or REST API, or **alter database DBS-NAME reset** in MaxL.

Createlocalconnection: Save a JDBC Connection

The `createlocalconnection` CLI command for Essbase creates a JDBC connection and stores it locally. To use this command, you need Service Administrator or power user role.

Description

A service administrator must use this command to create and save the local connection before anyone can use the CLI `dataload` or `dimbuild` commands with the streaming option. You must also set an environment variable `EXTERNAL_CLASSPATH` to point to the `.jar` file for your database driver (see Build Dimensions and Load Data by Streaming from a Remote Database).

Syntax

```
createLocalConnection [-verbose] -name streamConnection -connectionstring
connectionString -user userName [-driver jdbcDriver] [-password password]
```

Option	Abbreviation	Description
-verbose	-v	Show extended descriptions
-name	-N	Connection name
-connectionstring	-cs	JDBC connection string. Format can be with service name, as follows: <code>jdbc:oracle:thin:@host:port/service_name</code> or with SID, as follows: <code>jdbc:oracle:thin:@host:port:SID</code> The syntax formats above apply for Oracle Database. See Examples section for minor differences in the connection string syntax when you are working with other providers.
-user	-u	User name
-driver	-D	JDBC driver. If not provided, Oracle Database is considered the default, as <code>oracle.jdbc.driver.OracleDriver</code>
-password	-p	Password (optional)

If you have network connectivity between an external source of data and Essbase, it is most efficient to define application-level or global connections and Datasources in the Essbase web interface. These definitions help you to easily "pull" data from the external source. If you have no network connectivity between Essbase and the external source of data, then you can stream data loads or dimension builds using the CLI by first using this command to create a local connection, and then issuing the `dataload` or `dimbuild` command with the stream option.

Notes

After migrating to Release 21.4 or higher, the Service Administrator needs to recreate any saved local connections that were created using this command in a previous release.

Examples

- [Oracle DB - Service Name](#)
- [Oracle DB - SID](#)
- [DB2](#)
- [MySQL](#)
- [Microsoft SQL Server](#)
- [Teradata](#)

Oracle DB - Service Name

If the `-driver` option and `jdbcDriver` parameter are not provided, Oracle database is the assumed database by default.

```
esscs createLocalConnection -N OracleDBConnection2 -cs
jdbc:oracle:thin:@host1.example.com:1521/ORCL.esscs.host1.oraclecloud.com -u
OracleUser
```

Oracle DB - SID

If the `-driver` option and `jdbcDriver` parameter are not provided, Oracle database is the assumed database by default.

```
esscs createLocalConnection -N OracleDBConnection1 -cs
jdbc:oracle:thin:@myhostname01:1521:ORCL -u OracleUser -D
oracle.jdbc.driver.OracleDriver
```

DB2

If the `-driver` option and `jdbcDriver` parameter are not provided, Oracle database is the assumed database by default.

```
esscs createLocalConnection -N DB2conn -cs jdbc:db2://
myhostname02.example.com:50000/TBC -u myDB2User -D com.ibm.db2.jcc.DB2Driver
```

MySQL

If the `-driver` option and `jdbcDriver` parameter are not provided, Oracle database is the assumed database by default.

```
esscs createLocalConnection -N MySQLconn -cs jdbc:mysql://
myhostname03.example.com:3306/tbc -u MySQLUsr -D com.mysql.jdbc.Driver
```

Microsoft SQL Server

If the `-driver` option and `jdbcDriver` parameter are not provided, Oracle database is the assumed database by default.

```
esscs createLocalConnection -N MSSQLConn -cs jdbc:sqlserver://
myhostname04.example.com:1433 -u MSSQLUsr -D
com.microsoft.sqlserver.jdbc.SQLServerDriver
```

Teradata

If the `-driver` option and `jdbcDriver` parameter are not provided, Oracle database is the assumed database by default.

```
esscs createLocalConnection -N TeraDconn -cs jdbc:teradata://
myhostname05.example.com/DBS_PORT=1025 -u TeraUsr -D
com.teradata.jdbc.TeraDriver
```

Dataload: Load Data to a Cube

The dataload CLI command for Essbase loads data to a cube. To use this command, you need at least Database Update permission.

This command requires one of the following sets of options:

- Data file and optional rule file
- Rule file with user name and password
- Stream option referencing a saved local connection

The source database should be accessible within the client network, as not all database drivers can work with Java proxies.

To load data, you must first upload the data load and rule files to the cube directory. You can use the CLI to upload files. See [Upload: Add Cube Files](#).

Syntax

```
dataload [-verbose] -application appname -db cubename -file filename [| -
catalogfile catalogFile] [-rule rulesFile | -catalogrulefile
catalogRulesFile] [-user username [-password password]] [-stream] [-
connection connectionName][-query queryString] [-rows n] [-abortOnError]
```

Option	Abbreviation	Description
-verbose	-v	Show extended descriptions
-application	-a	Application name
-db	-d	Database (cube) name
-file	-f	Data load file name. You do not need to give a full path. Files are assumed to be in the relevant database directory. You can use <code>-catalogfile</code> in place of this option.

Option	Abbreviation	Description
-rule	-r	Optional. Rule file name. You do not need to give a full path. Files are assumed to be in the relevant database directory. You can use <code>-catalogrulefile</code> in place of this option.
-catalogfile	-CF	Data load file name from the file catalog. You can use this option in place of <code>-file</code> .
-catalogrulefile	-CRF	Rule file name from the file catalog. You can use this option in place of <code>-rule</code> .
-user	-u	Optional. User name. Requires password if used. If you are using a saved connection and Datasource, no user name and password are required. If you are not using a saved connection, and the rule file connects to an RDBMS, specify the user name and password to connect to the RDBMS.
-password	-p	Optional. Password for user. If omitted, user will be prompted for password.
-stream	-S	Optional. Use streaming data load. Requires <code>-conn</code> option if used.
-connection	-conn	Required if streaming option is used. Name of a saved connection that was created using the createlocalconnection CLI command.
-query	-q	Optional. Database query to submit along with the streaming data load.
-rows	-rows	Optional. Number of rows to stream simultaneously. Default is 100.
-abortOnError	-abort	Abort data load if error is encountered

Examples

```
esscs dataload -a Sample -db Basic -f Calcdat.txt -abort true
```

```
esscs dataload -a Sample -db Basic -r Basic.rul -S -conn oraConn -q "Select *
from Data" -rows 50
```

```
esscs dataload -a Sample -db Basic -CF /users/weblogic/Data_Basic.txt -r
Data.rul -abortonerror
```

```
esscs dataload -a Sample -db Basic -CF /users/weblogic/Data_Basic.txt -CRF /
shared/Data.rul -abort
```

```
esscs dataload -a Sample -db Basic -CRF /shared/Data.rul -S -conn
localConnectionName -q "Select * from Table"
```

You can also load data using Cube Designer, Jobs in the Essbase web interface or REST API, or **import data** in MaxL.

Deletefile: Remove Cube Files

The `deletefile` CLI command for Essbase removes cube artifacts from the application, database, or user home directory. To delete files from a cube, you need at least Database

Manager permission for the cube. No special permissions are required to delete files from your user directory.

Syntax

```
deletefile [-verbose] -file fileName [-application application [-db
database] [| -catalogfile catalogFile]]
```

Option	Abbreviation	Description
-verbose	-v	Show extended descriptions
-file	-f	Name of the file to delete
-application	-a	Optional. Application name. If not provided, files are assumed to be in your user home directory.
-database	-db	Optional. Database (cube) name
-catalogfile	-CF	File path and name from the file catalog. You can use this option in place of -file.

Examples

```
esscs deletefile -a Sample -d Basic -f Act1.rul
```

```
esscs deletefile -CF /shared/Data.txt
```

You can also manage files in Cube Designer, the Essbase web interface, or REST API.

Deploy: Create a Cube from a Workbook

The deploy CLI command for Essbase creates a cube from an Excel application workbook. To run this command, you need at least Power User role.

Syntax

```
deploy [-verbose] -file fileName [-application application [-database
database] | -catalogfile catalogFile] [-restructureoption restructureOption]
[-loaddata] [-recreateapplication] [-createfiles] [-executescript]
```

Option	Abbreviation	Description
-verbose	-v	Show extended descriptions
-file	-f	Name of the application workbook file
-application	-a	Optional. Application name. If not provided, application name will be taken from the workbook.
-database	-db	Optional. Database (cube) name. If not provided, database name will be taken from the workbook.
-catalogfile	-CF	Application workbook from the file catalog. You can use this option in place of -file.
-loaddata	-l	Optional. Load data, if the application workbook contains a data worksheet. Otherwise, only metadata is imported into the cube.

Option	Abbreviation	Description
-restructureoption	-R	Optional. Keyword indicating the desired restructuring option. The options for block storage cubes are: <ul style="list-style-type: none"> ALL_DATA—Preserve all data NO_DATA—Preserve no data LEAFLEVEL_DATA—Preserve level 0 (leaf level) data INPUT_DATA—Preserve input data The options for aggregate storage cubes are: <ul style="list-style-type: none"> ALL_DATA—Preserve all data NO_DATA—Preserve no data
-recreateapplication	-ra	Optional. Re-create the application, if it already exists
-createfiles	-cf	Optional. Create cube artifacts in the files directory in Essbase.
-executescript	-e	Optional. Execute calculation scripts. Applicable only if the application workbook contains a calculation worksheet with Execute Calc set to Yes in the definitions.

Examples

```
esscs deploy -v -a SampleD1 -d BasicD1 -f Sample_Basic.xlsx -l -ra -cf -e
```

```
esscs deploy -CF "/gallery/Applications/Demo Samples/Block Storage/Sample_Basic.xlsx" -a Sample1 -l -cf -e -R ALL_DATA
```

You can also deploy cubes using Cube Designer, or by using the Import option in the **Applications** section of the Essbase web interface.

Dimbuild: Load Dimensions to a Cube

The dimbuild CLI command for Essbase loads dimensions to a cube. To run this command, you need at least Database Manager permission for the cube.

Before you can load dimensions, you must first upload the dimension-build and rule files to Essbase. You can use the CLI to upload files. See [Upload: Add Cube Files](#).

Syntax

```
dimbuild [-verbose] -application appName -db cubeName -file fileName [| -catalogfile catalogFile] -rule rulesFile [| -catalogrulefile catalogRulesFile] [-user userName [-password password]] [-stream] [-connection connectionName] [-query queryString] [-rows n] [-restructureOption restructureOption] [-forcedimbuild]
```

Option	Abbreviation	Description
-verbose	-v	Show extended descriptions
-application	-a	Application name
-db	-d	Database (cube) name

Option	Abbreviation	Description
-file	-f	Dimension build file name. You do not need to give a full path. Files are assumed to be in the relevant application or database directory. You can use -catalogfile in place of this option.
-rule	-r	Rule file name. You do not need to give a full path. Files are assumed to be in the relevant application or database directory. You can use -catalogrulefile in place of this option.
-catalogfile	-CF	Dimension build file name from the file catalog. You can use this option in place of -file.
-catalogrulefile	-CRF	Rule file name from the file catalog. You can use this option in place of -rule.
-user	-u	Optional. User name. Requires password if used. If you are using a saved connection and Datasource, no user name and password are required. If you are not using a saved connection, and the rule file connects to an RDBMS, specify the user name and password to connect to the RDBMS.
-password	-p	Optional. Password for user. If omitted, user will be prompted for password.
-stream	-S	Optional. Use streaming dimension build. Requires -conn option if used.
-connection	-conn	Required if streaming option is used. Name of a saved connection that was created using the createlocalconnection CLI command.
-query	-q	Optional. Database query to submit along with the streaming dimension build.
-rows	-rows	Optional. Number of rows to stream simultaneously. Default is 100.
-restructureOption	-R	Controls your preservation choices for the outline restructure. For block storage, possible options are: <ul style="list-style-type: none"> • ALL_DATA: Preserve all data when loading dimensions. • NO_DATA: Do not preserve data. • LEAFLEVEL_DATA: Preserve only level 0 data values. If all data required for calculation resides in level-0 members, then you should select this option. All upper-level blocks are deleted before the cube is restructured. When the cube is recalculated, the upper-level blocks are re-created. • INPUT_DATA: Preserve only input data. For aggregate storage, possible options are: <ul style="list-style-type: none"> • ALL_DATA: Preserve all data when loading dimensions. • NO_DATA: Do not preserve data.
-forcedimbuild	-F	Continue the dimension build even if other user activities are in progress. This cancels active user sessions.

Examples

```
esscs dimbuild -a Sample -d Basic -r Basic.rul -u smith -p password -R
NO_DATA -F
```

```
esscs dimbuild -a Sample -d Basic -r Basic.rul -S -conn oraConn -q "Select *
from Data" -rows 50 -R NO_DATA
```

```
esscs dimbuild -a Sample -db Basic -CRF /users/weblogic/Dim_Market.rul -CF /
shared/Market.txt -R ALL_DATA -F
```

You can also load dimensions using Cube Designer, Jobs in the Essbase web interface or REST API, or **import dimensions** in MaxL.

Download: Get Cube Files

The download CLI command for Essbase downloads cube artifacts from an instance of Essbase to a local directory.

You may need to download text files, rule files, or calculation script files from a cube, so you can work on them or upload them to another cube. To download cube artifacts, you need at least Database Update permission.

Syntax

```
download [-verbose] -file filename [ | -catalogfile catalogFile] [-application
appName [-db cubeName]] [-localdirectory path] [-overwrite] [-nocompression]
```

Option	Abbreviation	Description
-verbose	-v	Show extended descriptions
-file	-f	Name of file to download
-application	-a	Optional. Application name. If not provided, artifacts are downloaded from your user home directory.
-db	-d	Optional. Database (cube) name
-catalogfile	-CF	File in the file catalog. You can use this option in place of -file.
-localdirectory	-ld	Optional. A local directory path
-overwrite	-o	Optional. Overwrite existing file
-nocompression	-nc	Optional. Disable compression of data transfer

Examples

```
esscs download -v -f Product003.rul -a Sample -d Basic -ld c:/temp -o
```

```
esscs download -f Acli.rul -ld c:/temp -o
```

```
esscs download -CF /shared/Acli.rul -ld c:/temp -o
```

You can also manage files in Cube Designer, the Essbase web interface, or REST API.

Help: Display Command Syntax

The help CLI command for Essbase displays command-level help in the console or terminal.

Syntax

```
[command] -help | -h
```

Examples

```
esscs -help
```

```
esscs -h
```

```
esscs dataload -help
```

LcmExport: Back Up Cube Files

The lcmexport CLI command for Essbase backs up applications and cube artifacts to a Lifecycle Management (LCM) .zip file, which it downloads to your local machine. To run this command, you need at least Application Manager permission.

Syntax

```
lcmExport [-verbose] -application appname|-allApp -zipfilename filename [-localDirectory path][-threads threadscount][-skipdata][-overwrite][-generateartifactlist][-include-server-level][-cube][-exportdata][-filetype][-exportpartitions][-exportfilters][-restEncryPassword]
```

Option	Abbreviation	Description
-verbose	-v	Optional. Show extended descriptions.
-application	-a	Name of application to back up.
-allApp	-aa	Optional (and case-sensitive). If used instead of -application, exports all applications to a single zip file. lcmimport can accept single-application zip files or multiple-application zip files.
-zipfilename	-z	Optional. Name of compressed file to hold backup files.
-localdirectory	-ld	Optional. A local directory path. If not specified, the zip is saved in <i><Application Directory>/catalog/users/<user_name></i> on the Essbase server.
-threads	-T	Optional. Number of threads to spawn if using parallel export. Minimum: 10
-skipdata	-skip	Optional. Do not include data in the backup.
-overwrite	-o	Optional. Overwrite existing backup file.

Option	Abbreviation	Description
-generateartifactlist	-gal	Optional. Generate a text file containing a complete list of the exported artifacts. You can use this text file to manage the import of artifacts. For example, you can rearrange the order of artifacts in the list to control the order in which they are imported. You can skip importing some artifacts by removing or commenting out items in the list.
-include-server-level	-isl	Optional. Include globally defined connections and Datasources.
-cube	-c	Optional. Export a single cube. This option can be specified along with the options to export only: data, files of certain types, partitions, or filters.
-exportdata	-d	Optional. Only export data.
-filetype	-ft	Optional. Only export files of the specified type. Supported file types include OTL (outline), TXT (text), RUL (rule), CSC (calc script), DTR (drill through report definition), and Excel (only .xls files are exported. No .xlsx files are exported). Examples: <pre>esscs lcmexport -a sample -z sampleXLSOnly.zip -v -ft excel</pre> <pre>esscs lcmexport -a sample -z sampleTXTOnly.zip -v -ft txt</pre>
-exportpartitions	-ep	Optional. Only export partition definitions. Lifecycle Management (LCM) import operations (and Migration Utility import) are not supported for migration of federated partitions. Federated partitions must be recreated manually on the target.
-exportfilters	-ef	Optional. Only export security filters.
-restEncryPassword	-encryPwd	If the application is encrypted, a password to protect the encrypted application during migration. The password must be between 6-15 characters, and should not contain any of the following special characters: ?=.,*!@#&() [{}];;' / ~\$^+<>- Caution: If this password is forgotten, there is no way to retrieve it, and the application cannot be imported.

Notes

This command, like other CLI commands, can be used from outside the Essbase machine, whereas the LCM utility must be run on the Essbase machine.

Example

```
esscs lcmExport -v -a Sample -z Sample.zip -ld c:/temp -skip -o -gal -isl
```

Windows Script Example

The following Windows script, `lcmexportall.bat`, exports all applications to the current local directory from which CLI was called.

```

set ESSCLI_ID=%USERNAME%_%random%
@echo on
echo Login to Essbase
call esscs login -u myusername -p mYpa55w0rD -url https://
myserver.example.com:9000/essbase
echo Export all apps and download to this directory
call esscs lcmexport -aa -z allapps.zip
echo Log out of Essbase
call esscs logout
@echo off

```

LcmImport: Restore Cube Files

The `lcmimport` CLI command for Essbase restores cube artifacts from a Lifecycle Management (LCM) `.zip` file. To run this command, you must be the power user who created the application, or a service administrator.

Syntax

```

lcmImport [-verbose] -zipfilename filename [-overwrite] [-targetappName
targetApplicationName] [-include-server-level] [-artifactlist artifactList] [-
restEncryPassword]

```

Option	Abbreviation	Description
-verbose	-v	Optional. Show extended descriptions
-zipfilename	-z	Name of compressed file containing backup files
-overwrite	-o	Optional. Recreate the target application.
-targetappName	-ta	Optional. Target application name, if you want it to be different from the source name.

Option	Abbreviation	Description
-artifactlist	-al	<p>Optional. Name of the file containing the list of artifacts to import. This file can be generated from lcmexport. To skip artifacts, comment out or delete entries from the list. For example, to skip importing audit records, comment out that line, as shown:</p> <pre># -----IMPORT----- import @Provisions import @Databases/Basic #import @Databases/Basic/Audit import @Databases/Basic/Text_files import @Databases/Basic/Xml_files import @Databases/Basic/Calc_scripts import @Databases/Basic/Open_XML_Excel_files import @Databases/Basic/ScenarioManagement import @Databases/Basic/Provisions import @Databases/Basic/Rule_files</pre> <p>To control import order, rearrange the <code>import</code> entries in the text file.</p> <p>If <code>–overwrite</code> is used, the import operation deletes and recreates the entire application, importing only the artifacts present in the list. If <code>–overwrite</code> is not used, the import operation includes the artifacts specified in the list, without impacting any other artifacts already present in the target application.</p>
-include-server-level	-isl	Optional. Include globally defined connections and Datasources.
-restEncryPassword	-encryPwd	<p>If the application is encrypted, a password to protect the encrypted application during migration. The password must be between 6-15 characters, and should not contain any of the following special characters: <code>?=.,*!@#&() [{}];;/~\$^+<>-</code></p> <p>Caution: If this password is forgotten, there is no way to retrieve it, and the application cannot be imported.</p>

Notes

- This command, like other CLI commands, can be used from outside the Essbase machine, whereas the LCM utility must be run within the Essbase machine.
- After the LCM import completes, you may need to take further action to restore migrated connections to external sources. To do this, open the connection and enter the password.
- When partitions exist between cubes being migrated, you must import the data source before the data target. Otherwise, partition definitions may not be restored.

Lifecycle Management (LCM) import operations (and Migration Utility import) are not supported for migration of federated partitions. Federated partitions must be recreated manually on the target.

- LCM Import does not migrate location alias credentials. You must replace your location alias credentials, either by recreating location aliases using MaxL, or by editing the location alias credentials in the XML exported by LCM Export.

Example

```
esscs lcmImport -z C:/Sample/Sample.zip -o -al C:/Sample/Sample.txt
```

Listapp: Display Applications

The listapp CLI command lists applications that you have access to on this instance of Essbase.

Syntax

```
listapp [-verbose] [-details]
```

Option	Abbreviation	Description
-verbose	-v	Optional. Show extended descriptions
-details	-dtl	Optional. Display more details in the output (application type and current status).

Example

```
esscs listapp -v -dtl
```

Listdb: Display Cubes

The listdb CLI command lists databases that you have access to within a specified Essbase application.

Syntax

```
listdb [-verbose] -application applicationName [details]
```

Option	Abbreviation	Description
-verbose	-v	Optional. Show extended descriptions
-application	-a	Application name
-details	-dtl	Optional. Display status details in the output

Example

```
esscs listdb -v -a Sample -dtl
```

Listfiles: Display Files

The listfiles CLI command lists cube artifacts that exist on an instance of Essbase.

Cube artifacts may include data files, workbooks, rule files, calculation script files, or other artifacts. Cube artifacts include any files that are needed to perform actions on applications and cubes.

To list files for a cube, you need at least Database Access permission for the application. No special permissions are required to list files from your user directory.

Syntax

```
listfiles [-verbose] [-type filetype] [-application appname [-db cubename] | -catalogpath catalogPath]
```

Option	Abbreviation	Description
-verbose	-v	Optional. Show extended descriptions
-type	-t	Optional. File extension/type to display, not including the period. Supported file types are: <ul style="list-style-type: none"> • .csc (calculation scripts) • .rul (rule files) • .txt (text files) • .msh (MaxL scripts) • .xls, .xlsx (Excel workbooks) • .xlsm (macro-enabled Excel workbooks) • .xml (XML files) • .zip (compressed zip files) • .csv (comma-separated files)
-application	-a	Optional. Application name. If not provided, files from your user home directory are displayed.
-db	-d	Optional. Database (cube) name
-catalogpath	-CP	Optional. Catalog path to the filename. Can be used instead of -a [-d] to specify the catalog location of the file(s).

Examples

```
esscs listfiles -t rul -a Sample -d Basic
```

```
esscs listfiles -CP "/shared"
```

You can also manage files in Cube Designer, the Essbase web interface, or REST API.

Listfilters: View Security Filters

The listfilters CLI command displays a list of Essbase security filters. You need at least Database Manager permission on the application to see filters for any cubes in the application.

Syntax

```
listfilters [-verbose] -application appname -db cubename
```

Option	Abbreviation	Description
-verbose	-v	Optional. Show extended descriptions
-application	-a	Application name
-db	-d	Database (cube) name

Example

```
esscs listfilters -v -a Sample -d Basic
```

Listlocks: View Locks

The listlocks CLI command for Essbase displays any locked data blocks or cube-related objects. To run this command, you need at least Database Access permission on the application.

Syntax

```
listlocks [-verbose] -application appname -db cubename [-object]
```

Option	Abbreviation	Description
-verbose	-v	Optional. Show extended descriptions
-application	-a	Application name
-db	-d	Database (cube) name
-object	-obj	Optional. Display locked files/artifacts.

Example

```
esscs listlocks -v -a Sample -d Basic -obj
```

Listvariables: Display Substitution Variables

The listvariables CLI command for Essbase lists substitution variables defined at the cube, application, or global scope. You need at least Database Access permission to see variables for a cube, Application Manager role to see variables for an application, and Service Administrator role to see global variables.

Syntax

```
listvariables [-verbose] [-application application [-db database]]
```

Option	Abbreviation	Description
-verbose	-v	Show extended descriptions.
-application	-a	Optional. Application name.
-database	-db	Optional. Database (cube) name.

Examples**Cube level**

```
esscs listvariables -a Sample -db Basic
```

Application level

```
esscs listvariables -a Sample
```

Global level

```
esscs listvariables
```

Setpassword: Store CLI Credentials

The setpassword CLI command for Essbase stores a password associated with your client/user combination. In subsequent sessions, you can log in without entering a password.

Syntax

```
setpassword [-verbose] -essbaseurl URL -user userName
```

Option	Abbreviation	Description
-verbose	-v	Optional. Show extended descriptions
-essbaseurl	-url	Address of an instance of Essbase
-user	-u	Your user name

Notes

After migrating to Release 21.4 or higher, you must reset any stored passwords that were saved using this command in a previous release.

Example

```
esscs setpassword -url https://myEssbase-test-  
myDomain.analytics.us2.example.com/essbase -user rschmidt
```

Start: Start an Application or Cube

The start CLI command starts an Essbase application or cube, loading it into memory. To run this command, you need at least Database Access permission on the application.

Syntax

```
start [-verbose] -application appname [-db cubename]
```

Option	Abbreviation	Description
-verbose	-v	Optional. Show extended descriptions
-application	-a	Application name
-db	-d	Optional. Database (cube) name

Example

```
esscs start -v -a Sample -d Basic
```

Stop: Stop an Application or Cube

The stop CLI command stops an Essbase application or cube. To run this command, you need at least Database Access permission on the application.

Syntax

```
stop [-verbose] -application appname [-db cubename]
```

Option	Abbreviation	Description
-verbose	-v	Optional. Show extended descriptions
-application	-a	Application name
-db	-d	Optional. Database (cube) name

Example

```
esscs stop -v -a Sample -d Basic
```

Unsetpassword: Remove Stored CLI Credentials

The unsetpassword CLI command for Essbase removes stored login credentials associated with your client/user combination, reversing the effect of setpassword.

Syntax

```
unsetpassword [-verbose] -essbaseurl URL -user userName
```

Option	Abbreviation	Description
-verbose	-v	Show extended descriptions
-essbaseurl	-url	Address of an instance of Essbase
-user	-u	The user whose password to unset

Example

```
esscs unsetpassword -url https://myEssbase-test-  
myDomain.analytics.us2.example.com/essbase -u user1
```

Upload: Add Cube Files

The upload CLI command uploads cube artifacts from a local directory to an instance of Essbase.

To perform tasks such as data loads, dimension builds, calculations, or other operations, you may need to upload data files, rule files, calculation script files, or other artifacts to the cube directory. You can also upload the artifacts to your user directory.

To upload files to a cube, you need at least Database Manager permission. No special permissions are required to upload to your user directory.

 **Note:**

You can enable antivirus scanning in the Essbase web interface so that files are scanned for viruses before they are uploaded to the server.

Syntax

```
upload [-verbose] -file filename [-application appname [-db cubename] | -
catalogpath catalogPath] [-overwrite] [-nocompression][-compressionalgorithm]
```

Option	Abbreviation	Description
-verbose	-v	Optional. Show extended descriptions
-file	-f	Name of file to upload
-application	-a	Optional. Application name. If not provided, files are uploaded to your user directory, or to the catalog path specified in <i>-CP</i> .
-db	-d	Optional. Database (cube) name. Requires <i>-a</i> .
-catalogpath	-CP	Optional. Catalog path to the filename. Can be used instead of <i>-a</i> [<i>-d</i>] to specify the catalog location of the file.
-overwrite	-o	Optional. Overwrite existing file
-nocompression	-nc	Optional. Disable compression of data transfer
-compressionalgorithm	-ca	Optional. Available if <i>-nc</i> is not used. Defines which compression algorithm to use for data transfer. Possible choices: gzip or lz4 . <ul style="list-style-type: none"> • gzip—Default if compression is used. Provides smaller data transfer with slower calculation. • lz4—Provides faster calculation with a slower data transfer. Usage examples: <pre>-ca gzip</pre> <pre>-ca lz4</pre>

 **Note:**

File extensions must be lower case. For example, *filename.txt*.

Examples

```
esscs upload -v -f c:/temp/Max101.msh -a Sample -d Basic -o -ca lz4
```

```
esscs upload -f C:/temp/Act1.rul -CP /shared
```

You can also manage files in Cube Designer, the Essbase web interface, or REST API.

Version: Display API Version

The version CLI command gets the version of the REST API that is associated with this instance of Essbase.

Syntax

```
version
```

Example

```
esscs version
```


3

MaxL

Using MaxL, you can administer and query Essbase through a scripting language.

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 begin with a verb, such as Create, Alter, and Display.

- [Manage Essbase Using the MaxL Client](#)
- [How to Read MaxL Railroad Diagrams](#)
- [MaxL Statements](#)
- [MaxL Definitions](#)
- [MaxL Shell Commands](#)
- [Reserved Words List](#)
- [MaxL BNF](#)
- [MaxL Statements \(Aggregate Storage\)](#)
- [MaxL Use Cases](#)

Manage Essbase Using the MaxL Client

To communicate with Essbase using MaxL scripts or statements, use the MaxL Client to issue the statements over HTTP or HTTPS.

- [Prerequisites to Set Up the MaxL Client](#)
- [Download and Use the MaxL Client](#)

If you want to run MaxL statements on the Essbase Server rather than from a client, connect to the server and run the MaxL startup script, `startMAXL.sh` or `startMAXL.bat`. The script is located in `<Domain Root>/<Domain Name>/esstools/bin`. If you do not know where that is in your Essbase Server, refer to Environment Locations in the Essbase Platform.

Prerequisites to Set Up the MaxL Client

Before you can use the MaxL Client, you will need the Essbase URL, and you may need to set up the TLS (SSL) certificate.

To run MaxL scripts or statements, you must be a power user or administrator. To prepare for using the MaxL Client,

1. Get the URL for the Essbase instance from your Service Administrator. Its basic format is:

```
https://IP-address:port/essbase
```

2. Using a web browser or cURL, test that you can reach the discovery URL from the client host. A discovery URL is the URL provided by your Service Administrator, with `/agent`

appended to the end. Here is a cURL example (for secure/TLS mode in an independent Essbase deployment):

```
curl https://192.0.2.1:9001/essbase/agent --tlsv1.2
```

Here is an example for a stack deployment of Essbase on OCI:

```
curl https://192.0.2.1:443/essbase/agent --tlsv1.2
```

If you have connectivity, you should see a response:

```
<html>
<head><title>Oracle&#x00ae; Essbase</title></head>
<body>
<H2>Oracle&#x00ae; Essbase</H2>
</body></html>
```

3. Set up the SSL certificate, if applicable to your organization.

- If you're using one of these deployment types, a Trusted CA Signed SSL Certificate is included:
 - Oracle Analytics Cloud
 - Oracle Analytics Cloud with Identity Cloud Service (IDCS) and Load Balancing
 - Cloud at Customer with Load Balancing
- If you're using Oracle Analytics Cloud or Cloud at Customer with LDAP (without Load Balancing), use a self-signed certificate.
- To check if a certificate is trusted, paste the discovery URL into a web browser. If **https** is green or a label says "Secure," it is trusted. If **https** is red or a label says "Not secure," it is untrusted.
- If you're using the MaxL Client in Essbase 21c with a self-signed certificate, you will have two options (do this after you download the client):

- a. Disable peer verification by setting the environment variable `API_DISABLE_PEER_VERIFICATION=1`

Linux example

Edit `startMAXL.sh`, adding the following line:

```
export API_DISABLE_PEER_VERIFICATION=1
```

Windows example

Edit `startMAXL.bat`, adding the following line:

```
set API_DISABLE_PEER_VERIFICATION=1
```

- b. Import the self-signed certificate to the client trust store (`cacert.pem`) and set the environment variable `API_CAINFO=CA <certificate file path>`. The client verifies the server's digital certificate using a provided ca-bundle certificate store. Provide the ca-bundle location by specifying the environment variable `API_CAINFO=CA <certificate file path>`

Linux example

Edit `startMAXL.sh`, adding the following line:

```
export API_CAINFO=/u01/cacert.pem
```

Windows example

Edit `startMAXL.bat`, adding the following line:

```
set API_CAINFO=c:/cacert.pem
```

If you don't provide *certificate file path*, the Essbase Runtime Client will try to get `ca-bundle` from the default OpenSSL installation location (applicable for Linux and Macintosh).

A `cacert.pem` is available in the MaxL Client download zip. Another sample source is: <https://curl.haxx.se/docs/caextract.html>.

Download and Use the MaxL Client

To run the MaxL Client for use with Essbase, download the latest version from the Console, set the proxy if needed, run the startup script, and log in.

The Essbase MaxL Client enables you to use MaxL over HTTP or HTTPS. MaxL is an administrative, language-based interface for managing cubes and artifacts. Be sure you are using the latest client version provided in the Console, as older, previously downloaded versions may not work correctly.

To run MaxL statements, you must be a power user or an administrator. Before you download the MaxL Client, refer to [Prerequisites to Set Up the MaxL Client](#).

If you're a [federated](#) SSO user in Oracle Identity Cloud Service, logging in using MaxL or CLI is not supported. Federated SSO login requires a browser window. Create a native IAM or IDCS user, and use that instead to log in using MaxL or CLI.

1. In the Essbase web interface, click **Console**.
2. In the Console, go to **Desktop Tools**, then **MaxL Clients**.
3. Click the tile for the appropriate MaxL Client for your platform to begin the download.
4. Save the compressed `EssbaseMaxl` file to your local drive.
5. Extract the contents of the compressed file to a folder.
6. If you're using a proxy, you must set the correct proxy in the MaxL execution script, `startMAXL.bat` or `startMAXL.sh`. The following example, applicable for editing `startMAXL.sh` for UNIX, tells MaxL to use the designated proxy (`proxy.example.com`), but to bypass using a proxy for the specific destinations listed in the exception list (`127.0.0.1`, `localhost`, and `something.example.com`).

```
export https_proxy=http://proxy.example.com
export no_proxy=127.0.0.1,localhost,something.example.com
```

For Windows, `startMAXL.bat` can be edited similarly but with different syntax.

```
set proxy proxy-server="https://proxy.example.com" bypass-
list="127.0.0.1;localhost;*.example.com"
```

7. If you're using Essbase deployed on Oracle Cloud Infrastructure and are using a self signed certificate, you must disable peer verification in the MaxL execution script. **Caution:** this solution should be only temporary, until you can obtain a trusted CA certificate. Here is an example using **bash** (for `startMAXL.sh`):

```
export API_DISABLE_PEER_VERIFICATION=1
```

8. Run the `startMAXL` batch or shell script. A command prompt opens, the environment setup completes, and the MaxL Client starts up.
9. Log in by providing your credentials and the Essbase URL in the MaxL **login** statement. In the following example, the user logging in, `User5`, is from a federated MSAD directory and logging in to Essbase On-Premise.

```
login user User5 P855w0r$4 on "https://192.0.2.1:9001/essbase/agent";
```



Tip:

See MaxL Troubleshooting for On-Premise installations.

In the following example, the user logging in, `admin1@example.com` is an Identity Cloud Service administrator who was set as the initial Essbase administrator during Essbase stack deployment on Oracle Cloud Infrastructure. As the password is not entered in this example, the administrator will be prompted to provide it next. The URL is the **essbase_url** from the job outputs resulting from the stack deployment.

```
login admin1@example.com on "https://192.0.2.1/essbase";
```

Any Identity Cloud Service user provisioned to work with Essbase can log in to MaxL, as long as they are provisioned as a power user or administrator.

10. Execute an interactive MaxL statement.

For example:

```
display database all;
```

To learn more about MaxL, see MaxL Statement Reference.

How to Read MaxL Railroad Diagrams

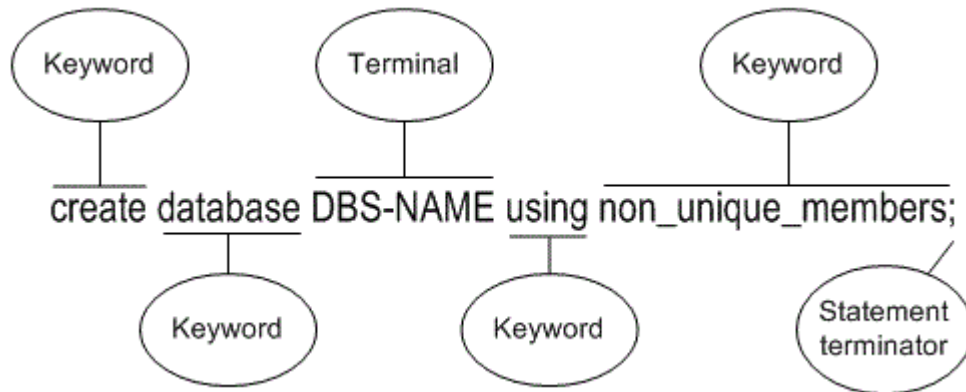
The MaxL grammar for Essbase is illustrated using a railroad syntax notation. The railroad diagrams illustrate all the valid (grammatically correct) statements that can be parsed by MaxL.

- [Anatomy of MaxL Statements](#)
- [Railroad Diagram Symbols](#)
- [Sample Railroad Diagram](#)

Anatomy of MaxL Statements

MaxL statements for Essbase are syntactically comprised of keywords and terminals. Understanding how they work together can help you build your MaxL script-writing skills.

- A **keyword**, 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, `DBS-NAME` is a terminal. Terminals need to be replaced with a valid name; for example, `sample.basic`.

Keywords cannot be used as terminals, unless enclosed in single quotation marks. For example, to create a database named database, the statement `create database database;` would return an error, but `create database "database";` would work.
- The semicolon indicates the end of a statement. Omitting a semicolon, or placing one before the expected end of a statement, results in a syntax error.
- A non-terminal, represented in upper-case with angle brackets `<>`, is defined in an additional diagram, usually below the main diagram. No non-terminal is shown here.

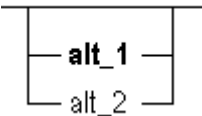
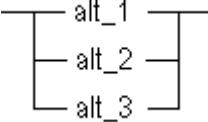
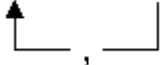
Railroad Diagram Symbols

The following table describes the meaning of symbols used in railroad diagrams.

Table 3-1 Railroad Diagram Symbols

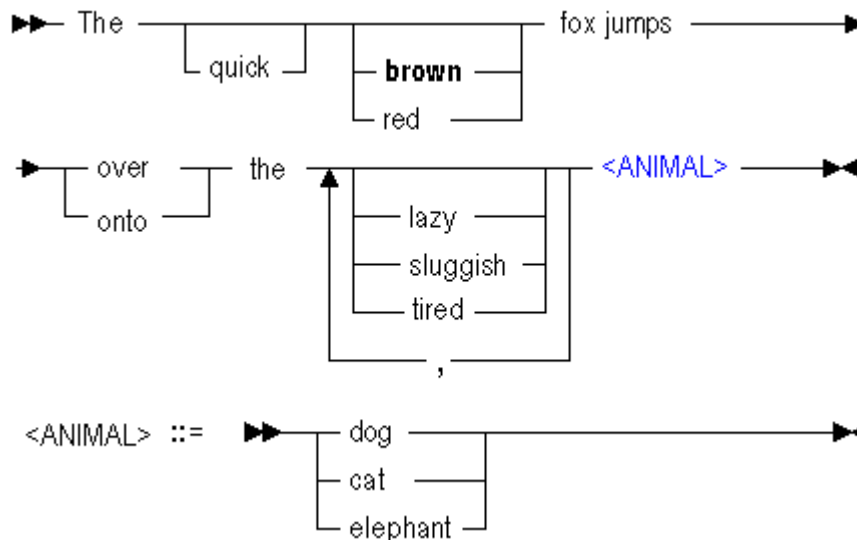
Symbol	Definition
▶▶	Statement begins here.
→	Statement continues on next line.
▶	Statement is continued from previous line.
▶▶	Statement ends here.

Table 3-1 (Cont.) Railroad Diagram Symbols

Symbol	Definition
	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."



Keywords and variables on the main line (with arrow markings) are required; optional grammar is recessed (lower than the main line). A vertical stack of words represents alternatives. Bold words indicate defaults when no word is chosen.

Valid sentences parse-able by the example grammar may include:

- 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 Statements

MaxL is a data-definition language for Essbase that has an expressive grammar you use to write statements. Statements help you perform administrative actions in Essbase.

In Essbase documentation, the syntax for MaxL DDL is illustrated using [railroad diagrams](#).

MaxL grammar is case-insensitive. Semicolon statement-terminators are required when using the MaxL Shell.

Keywords 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](#).

Topics covered in this section:

- [Listed By Verbs](#)
- [Listed by Objects](#)
- [MaxL Statement Reference](#)
- [Performance Statistics in MaxL](#)

Listed By Verbs

MaxL data-definition language for Essbase has statements beginning with the following verbs.

[alter](#)

[create](#)

[display](#)

[drop](#)

[execute](#)

[export](#)

[grant](#)

[import](#)

[login](#)

[logout](#) (see [MaxL Shell Commands](#))

[query](#)

[refresh](#)

Alter

MaxL data-definition language for Essbase has the following statements that begin with the verb alter.

- application
- database
- drillthrough
- filter
- group
- object
- partition
- session
- system
- tablespace
- trigger
- user

Create

MaxL data-definition language for Essbase has the following statements that begin with the verb create.

- application
- calculation
- database
- drillthrough
- filter
- function
- group (EPM Shared Services mode only)
- location alias
- macro
- outline
- partition
- trigger
- user (EPM Shared Services mode only)

Display

MaxL data-definition language for Essbase has the following statements that begin with the verb display.

- application
- calculation
- database
- drillthrough
- filter
- filter row
- function
- location alias
- lock
- macro
- object
- partition
- privilege
- session
- system
- tablespace
- trigger
- trigger spool
- variable

Drop

MaxL data-definition language for Essbase has the following statements that begin with the verb drop.

- application
- calculation
- database
- drillthrough
- filter
- function
- group
- location alias

- [lock](#)
- [macro](#)
- [object](#)
- [partition](#)
- [trigger](#)
- [trigger spool](#)
- [user](#)

Execute

MaxL data-definition language for Essbase has the following statements that begin with the verb execute.

- [aggregate process](#) (aggregate storage)
- [aggregate selection](#) (aggregate storage)
- [aggregate build](#) (aggregate storage)
- [allocation](#) (aggregate storage)
- [calculation](#)
- [custom calculation](#) (aggregate storage)

Export

MaxL data-definition language for Essbase has the following statements that begin with the verb export.

- [data](#)
- [LRO](#)
- [outline](#)
- [query_tracking](#) (aggregate storage)

Grant

MaxL data-definition language for Essbase has a grant statement. Click the link for syntax details and uses.

- [Grant](#)

Import

MaxL data-definition language for Essbase has the following statements that begin with the verb import.

- [data](#)
- [dimensions](#)
- [lro](#)

[query_tracking](#) (aggregate storage)

Login

Before you can send MaxL statements from the MaxL Shell to Essbase, you must log in to an Essbase Server session.

As a prerequisite to using MaxL, follow the client setup instructions in [Manage Essbase Using the MaxL Client](#).

Note:

Before logging in to an Essbase Server session, you must start the MaxL Shell (see MaxL Invocation Summary, in [MaxL Shell Commands](#)). Or, you can start the MaxL Shell and log in at the same time (see `-l Flag: Login` at the same link).

```
login USER-NAME [identified by] PASSWORD [on HOST-NAME]
```

- [USER-NAME](#)
- [PASSWORD](#)
- [HOST-NAME](#)

Example

For an independent deployment:

```
login admin pa5sw0rd on "https://myserver.example.com:9001/essbase/agent";
```

For a stack deployment on OCI:

```
login admin pa5sw0rd on "https://192.0.2.1:443/essbase/agent";
```

Note:

The Essbase system administrator logging in must have Identity Domain Administrator and Security Administrator role in the confidential identity application.

Query

MaxL data-definition language for Essbase has the following statements that begin with the verb `query`.

[archive_file](#)

[database](#)

[application](#) (for aggregate storage)

[application](#) (for block storage)

Refresh

MaxL data-definition language for Essbase has the following statements that begin with the verb refresh.

[custom definitions](#)

[outline](#)

[replicated partition](#)

Listed by Objects

MaxL data-definition language for Essbase has statements that operate on the following objects.

[aggregate_build](#) (aggregate storage)

[aggregate_process](#) (aggregate storage)

[aggregate_selection](#) (aggregate storage)

[allocation](#) (aggregate storage)

[application](#)

[archive_file](#)

[calculation](#)

[data](#)

[database](#)

[dimensions](#)

[drillthrough](#)

[filter](#)

[function](#)

[group](#)

[location alias](#)

[lock](#)

[lro](#)

[macro](#)

[object](#)

[outline](#)

[partition](#)

[privilege](#)
[query_tracking](#) (aggregate storage)
[session](#)
[system](#)
[tablespace](#) (aggregate storage)
[trigger](#)
[trigger spool](#)
[user](#)
[variable](#)

Aggregate Build

MaxL data-definition language for Essbase has one statement that operates on the aggregate build object. Click the link for syntax details and uses.

[execute aggregate build](#)

Aggregate Process

MaxL data-definition language for Essbase has one statement that operates on the aggregate process object. Click the link for syntax details and uses.

[execute aggregate process](#)

Aggregate Selection

MaxL data-definition language for Essbase has one statement that operates on the aggregate selection object. Click the link for syntax details and uses.

[execute aggregate selection](#)

Allocation

MaxL data-definition language for Essbase has one statement that operates on the allocation object. Click the link for syntax details and uses.

[execute allocation](#)

Application

MaxL data-definition language for Essbase has the following statements that operate on the application object. Click a link for syntax details and uses.

[alter](#)
[create](#)
[display](#)
[drop](#)
[query](#) (for aggregate storage only)

Archive_file

MaxL data-definition language for Essbase has one statement that operates on the archive_file object. Click the link for syntax details and uses.

[query](#)

Calculation

MaxL data-definition language for Essbase has the following statements that operate on the calculation object. Click a link for syntax details and uses.

[create](#)

[display](#)

[drop](#)

[execute](#)

[execute custom](#) (aggregate storage)

Custom Definitions

MaxL data-definition language for Essbase has the following statements that operate on the custom definition object. Click a link for syntax details and uses.

[create function](#)

[create macro](#)

[display function](#)

[display macro](#)

[drop function](#)

[drop macro](#)

[refresh custom definitions](#)

Data

MaxL data-definition language for Essbase has the following statements that operate on the data object. Click a link for syntax details and uses.

[export](#)

[import](#)

Database

MaxL data-definition language for Essbase has the following statements that operate on the database object. Click a link for syntax details and uses.

[alter](#)

[create](#)

[display](#)

[drop](#)

[query](#)

Dimensions

MaxL data-definition language for Essbase has one statement that operates on the dimensions object. Click the link for syntax details and uses.

[import](#)

Drillthrough

MaxL data-definition language for Essbase has the following statements that operate on the drillthrough object. Click a link for syntax details and uses.

[alter](#)

[create](#)

[display](#)

[drop](#)

Filter

MaxL data-definition language for Essbase has the following statements that operate on the filter object. Click a link for syntax details and uses.

[alter filter](#)

[create filter](#)

[display filter](#)

[display filter row](#)

[drop filter](#)

Function

MaxL data-definition language for Essbase has the following statements that operate on the function object. Click a link for syntax details and uses.

[create](#)

[display](#)

[drop](#)

[refresh](#)

Group

MaxL data-definition language for Essbase has the following verbs/statements you can use to work on the **group** object. Click the links for syntax details and uses.

[alter](#)

[create](#)

[drop](#)

Location Alias

MaxL data-definition language for Essbase has the following statements that operate on the location alias object. Click a link for syntax details and uses.

[create](#)

[display](#)

[drop](#)

Lock

MaxL data-definition language for Essbase has the following statements that operate on the lock object. Click a link for syntax details and uses.

[display](#)

[drop](#)

LRO

MaxL data-definition language for Essbase has the following statements that operate on the lro object. Click a link for syntax details and uses.

[export](#)

[import](#)

Macro

MaxL data-definition language for Essbase has the following statements that operate on the macro object. Click a link for syntax details and uses.

[create](#)

[display](#)

[drop](#)

[refresh](#)

Object

MaxL data-definition language for Essbase has the following statements that operate on the object object. Click a link for syntax details and uses.

[alter](#)

[display](#)

[drop](#)

Outline

MaxL data-definition language for Essbase has the following statements that operate on the outline object. Click a link for syntax details and uses.

[create](#)

[refresh](#)

see also [Dimensions](#)

Partition

MaxL data-definition language for Essbase has the following statements that operate on the partition object. Click a link for syntax details and uses.

[alter](#)

[create](#)

[display](#)

[drop](#)

[refresh replicated](#)

[refresh outline](#) for outline synchronization

Privilege

MaxL data-definition language for Essbase has the following statements that operate on the privilege object. Click a link for syntax details and uses.

[display](#)

[grant](#)

Query_tracking

MaxL data-definition language for Essbase has the following statements that operate on the query_tracking object. Click a link for syntax details and uses.

[export](#) (aggregate storage)

[import](#) (aggregate storage)

Session

MaxL data-definition language for Essbase has the following statements that operate on the session object. Click a link for syntax details and uses.

[alter](#)

[display](#)

[alter system](#) to stop a session

System

MaxL data-definition language for Essbase has the following statements that operate on the system object. Click a link for syntax details and uses.

[alter](#)

[display](#)

Tablespace

MaxL data-definition language for Essbase has the following statements that operate on the tablespace object. Click a link for syntax details and uses.

[alter](#)

[display](#)

Trigger

MaxL data-definition language for Essbase has the following statements that operate on the trigger object. Click a link for syntax details and uses.

[alter](#)

[create or replace](#)

[display](#)

[drop](#)

Trigger Spool

MaxL data-definition language for Essbase has the following statements that operate on the trigger spool object. Click a link for syntax details and uses.

[display](#)

[drop](#)

User

MaxL data-definition language for Essbase has the following statements that operate on the user object. Click a link for syntax details and uses.

[alter](#) to revoke filters

[create](#) to add to Essbase external users that exist in EPM Shared Services

[drop](#) to clean up remnant accounts from Essbase

[grant](#) to assign permissions

Variable

MaxL data-definition language for Essbase has the following statements that operate on the variable object. Click a link for syntax details and uses.

[display variable](#)

To add, drop, or set substitution variables:

[alter application](#)

[alter database](#)

[alter system](#)

MaxL Statement Reference

MaxL statements available for Essbase are different depending on whether you are administering an aggregate storage cube or a block storage cube (including hybrid mode).

Ways to navigate MaxL documentation include:

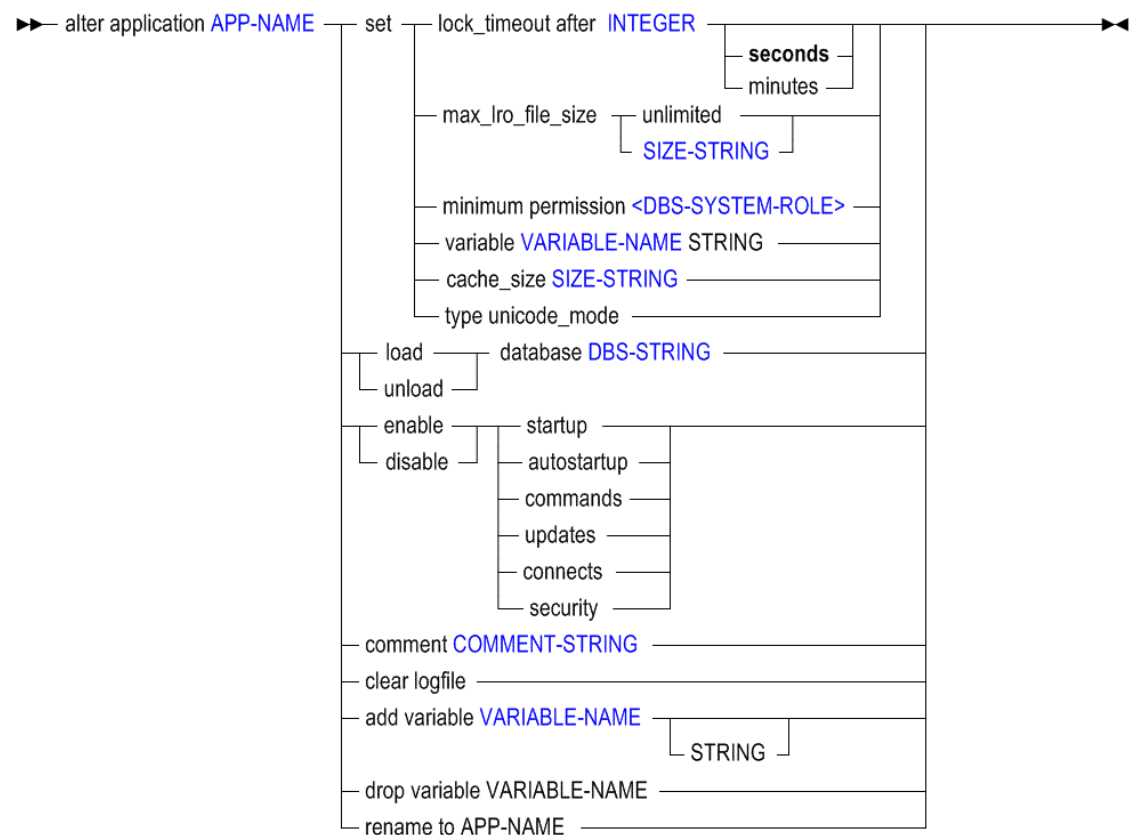
- [Listed By Verbs](#)
- [Listed by Objects](#)
- [Aggregate Storage List](#)

Alter Application

The MaxL alter application statement helps you change Essbase application-wide settings.

[Click here for aggregate storage version](#)

Syntax



- [APP-NAME](#)

- INTEGER
- SIZE-STRING
- DBS-SYSTEM-ROLE
- VARIABLE-NAME
- DBS-STRING
- COMMENT-STRING

Keywords

Use MaxL **alter application** to change the following application-wide settings. The minimum application permission required for most of the statements is Application Manager, with exceptions noted.

alter application APP-NAME set lock_timeout ...

Change the maximum time interval that locks on data blocks can be held by Smart View (or other grid clients') 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.

Example:

```
alter application Sample set lock_timeout after 30 minutes;
```

alter application APP-NAME 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.

Example:

```
alter application Sample set max_lro_file_size 450mb;
```

alter application APP-NAME 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.

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 APP-NAME 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.

Example:

```
alter application Sample 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 Sample application.

alter application APP-NAME set cache_size ...

Set the maximum size to which the application cache may grow. The application cache grows dynamically until it reaches this limit. The application cache is used for hybrid aggregation in block storage databases, and can help you manage memory usage for retrievals. This setting takes effect after you restart the application. To check the currently set limit, use the following MaxL statement:

```
query application APP-NAME get cache_size;
```

Example:

```
alter application Sample set cache_size 32mb;
```

alter application APP-NAME set type unicode_mode

Migrate an application to Unicode mode. Migration to Unicode mode cannot be reversed.

Example:

```
alter application Sample set type unicode_mode;
```

alter application APP-NAME load database ...

Start (by loading into memory) an idle database. You need at least Database Access permission provisioned in the application.

Example:

```
alter application Sample load database Basic;
```

alter application APP-NAME unload database ...

Stop (by unloading from memory) an active database. You need at least Database Access permission provisioned in the application.

Example:

```
alter application Sample unload database Basic;
```

alter application APP-NAME enable startup

Permit all users who have at least Database Access permission to load (start) the application. Startup is enabled by default.

Example:

```
alter application Sample enable startup;
```

alter application APP-NAME disable startup

Prevent all users from loading (starting) the application. Startup is enabled by default.

Example:

```
alter application Sample disable startup;
```

alter application APP-NAME enable autostartup

Start the application automatically when Essbase Server starts. By default, autostartup is disabled.

Example:

```
alter application Sample enable autostartup;
```

alter application APP-NAME disable autostartup

Do not start the application automatically when Essbase Server starts. By default, autostartup is disabled.

Example:

```
alter application Sample disable autostartup;
```

alter application APP-NAME 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.

Example:

```
alter application Sample enable commands;
```

alter application APP-NAME 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.

 **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.

By default, commands are enabled.

Example:

```
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 APP-NAME 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.

Example:

```
alter application Sample enable updates;
```

alter application APP-NAME 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.

Example:

```
alter application Sample disable updates;
```

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.

alter application APP-NAME 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.

Example:

```
alter application Sample enable connects;
```

alter application APP-NAME 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 shell 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.

Example:

```
alter application Sample disable connects;
```

alter application APP-NAME 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.

Example:

```
alter application Sample enable security;
```

alter application APP-NAME 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.

Example:

```
alter application Sample disable security;
```

alter application APP-NAME comment ...

Enter an application description (optional). The description can contain up to 80 characters.

Example:

```
alter application Sample comment 'Sales application';
```

alter application APP-NAME clear logfile

Delete the application log located in the application directory. A new log is created for entries recording subsequent application activity.

Example:

```
alter application Sample clear logfile;
```

alter application APP-NAME 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.

Example:

```
alter application Sample add variable Month Nov;
```

alter application APP-NAME drop variable

Remove a substitution variable and its corresponding value from the application.

Example:

```
alter application Sample drop variable Month;
```

alter application APP-NAME rename to

Rename the application. When you rename an application, the application and the application directory are renamed.

Example:

```
alter application Sample rename to Sample2;
```

Alter Database

The MaxL alter database statement helps you change Essbase database-wide settings.

[Click here for aggregate storage version](#)

Select a subset of **alter database** to view the syntax options:

- [Alter Database enable | disable](#)
- [Alter Database Set](#)

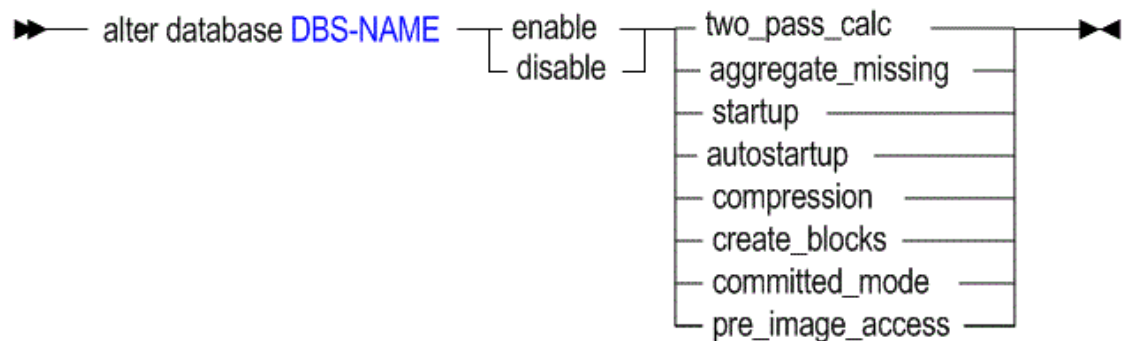
- [Alter Database \(Misc\)](#)

Alter Database enable | disable

The MaxL **alter database** statement with **enable|disable** keywords helps you turn on and off Essbase database-wide functionality.

[Click here for aggregate storage version](#)

Syntax



DBS-NAME

Keywords

Use MaxL **alter database** to change the following database-wide settings. The minimum application permission required for most of the statements is Database Manager, with exceptions noted.

alter database DBS-NAME enable two_pass_calc



Note:

Do not use two-pass calculation with hybrid mode cubes. Only use solve order.

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.

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.

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 CALC TWOPASS command in the calculation script.

Example:

```
alter database Sample.Basic enable two_pass_calc;
```

alter database DBS-NAME disable two_pass_calc

Do not recalculate database outline members tagged as Two Pass after a default calculation. Two-pass calculation is enabled by default.

Example:

```
alter database Sample.Basic disable two_pass_calc;
```

Prevents recalculation (after a default calculation) of members tagged as Two Pass.

alter database DBS-NAME enable aggregate_missing

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.

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.

Example:

```
alter database Sample.Basic enable aggregate_missing;
```

alter database DBS-NAME disable aggregate_missing

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.

Example:

```
alter database Sample.Basic disable aggregate_missing;
```

alter database DBS-NAME 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.

Example:

```
alter database Sample.Basic enable startup;
```

alter database DBS-NAME 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.

Example:

```
alter database Sample.Basic disable startup;
```

alter database DBS-NAME 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.

Example:

```
alter database Sample.Basic enable autostartup;
```

alter database DBS-NAME disable autostartup

Prevent automatic starting of the database when the application to which it belongs starts. Autostartup is enabled by default.

Example:

```
alter database Sample.Basic disable autostartup;
```

alter database DBS-NAME enable compression

Enable data compression. By default, Bitmap compression is enabled. To switch to a different compression type, use `alter database set compression`.

Example:

```
alter database Sample.Basic enable compression;
```

alter database DBS-NAME disable compression

Disable data compression. By default, Bitmap compression is enabled.

Example:

```
alter database Sample.Basic disable compression;
```

alter database DBS-NAME enable create_blocks

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.

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.

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:

```
2002Forecast = Actuals * 1.05;.
```

Example:

```
alter database Sample.Basic enable create_blocks;
```

alter database DBS-NAME disable create_blocks

Turn off the Create Blocks on Equation setting. The setting is disabled by default.

Example:

```
alter database Sample.Basic disable create_blocks;
```

alter database DBS-NAME enable committed_mode**Note:**

Committed access mode for a cube is no longer supported beginning in Essbase 21c.

Committed access means 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 `enable_pre_image_access` setting. The default isolation-level mode is Uncommitted, but Smart View and other grid clients' data-update operations are always in committed mode.

alter database DBS-NAME disable committed_mode

Turn off the Committed Mode setting, reverting to the default isolation level of Uncommitted for the database.

**Note:**

Smart View and other grid clients' data-update operations are always in committed mode.

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 `implicit_commit` settings.

alter database DBS-NAME enable pre_image_access

Committed access mode for a cube is no longer supported beginning in Essbase 21c. The following description is relevant only for releases before Essbase 21c.

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.

alter database DBS-NAME disable pre_image_access

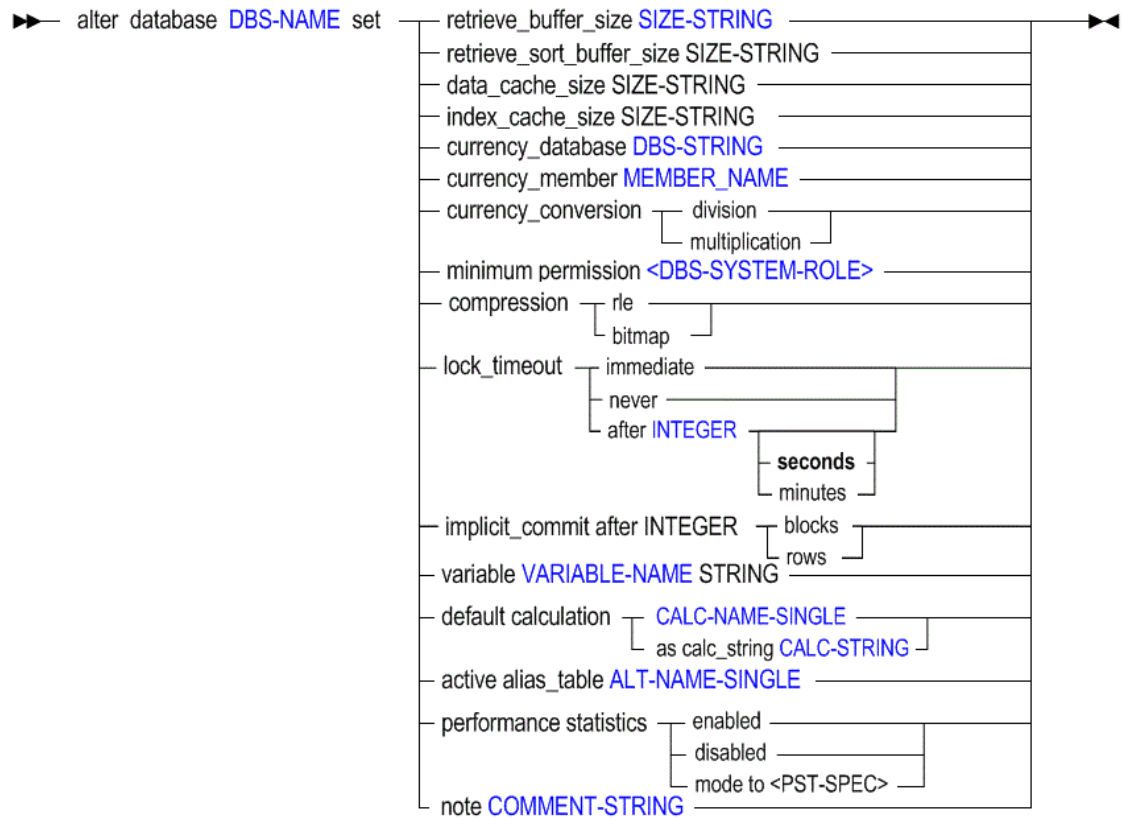
Disable pre-image access, disallowing read-only access to locked blocks of data at their last commit point (this setting is only applicable while the database is in committed mode). Pre-image access is enabled by default when the database is in committed mode.

Alter Database Set

The MaxL **alter database** statement under the **set** keyword helps you modify Essbase database-wide settings.

[Click here for aggregate storage version](#)

Syntax



- DBS-NAME
- SIZE-STRING
- DBS-STRING
- MEMBER-NAME
- DBS-SYSTEM-ROLE
- INTEGER
- VARIABLE-NAME
- CALC-NAME-SINGLE
- CALC-STRING
- ALT-NAME-SINGLE
- COMMENT-STRING

Keywords

Use MaxL **alter database set** to change the following database-wide settings. The minimum application permission required for most of the statements is Database Manager, with exceptions noted.

alter database DBS-NAME 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 20 KB. The minimum size is 2 KB. Increasing the size may improve retrieval performance.

Example:

```
alter database Sample.Basic set retrieve_buffer_size 20kb;
```

alter database DBS-NAME 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 20 KB. The minimum size is 2 KB. Increasing the size may improve retrieval performance.

Example:

```
alter database Sample.Basic set retrieve_sort_buffer_size 20kb;
```

alter database DBS-NAME set data_cache_size ...

Change the data cache size. The data cache is a buffer in block storage memory that holds uncompressed data blocks. You need at least Application Manager permission to set it. 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.

Example:

```
alter database Sample.Basic set data_cache_size 3072kb;
```

alter database DBS-NAME set index_cache_size ...

Change the index cache size. The index cache is a buffer in block storage memory that holds index pages. You need at least Application Manager permission to set it. When a data block is requested, Essbase looks at the index pages in the index cache to find its location on disk.

- Minimum index cache size is: 1 MB (1,048,576 bytes)
 - Maximum index cache size is: 256 TB
- Default index cache size is: 1 MB (1,048,576 bytes)

Example:

```
alter database Sample.Basic set index_cache_size 1mb;
```

alter database DBS-NAME set currency_database ...

Link the cube with a currency cube, enabling conversion of currency values from one currency into another.

Example:

```
alter database Sample_Currency.Internatl set currency_database Xchgrate;
```

alter database DBS-NAME set 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 main cube.

Example:

```
alter database Sample_Currency.Internatl set currency_member 'Act xchg';
```

alter database DBS-NAME set currency_conversion

Specify the method of currency conversion: to multiply, or divide, the exchange rates with the main cube. Indicate using keyword **multiplication** or **division**.

Example:

```
alter database Sample_Currency.Intern1 set currency_conversion
multiplication;
```

alter database DBS-NAME 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. This statement is supported for use in EPM Shared Services security mode only.

Example:

```
alter database Sample.Basic set minimum permission no_access;
```

alter database DBS-NAME set 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.

Example:

```
alter database Sample.Basic set compression rle;
```

alter database DBS-NAME set 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.

Example:

```
alter database Sample.Basic set compression bitmap;
```

alter database DBS-NAME set 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:

Smart View and other grid clients' data-update operations are always in committed mode.

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 DBS-NAME set 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).

The default frequency, if unspecified, is 3000, and may adjust dynamically during a calculation.

If Essbase Server runs on Oracle Exalytics In-Memory machine, for calculation and data load requests, the commit happens at the end of the command or request, and the default interval of 3000 (or any other value you specify) is ignored.

Example:

```
alter database Sample.Basic set implicit_commit after 3000 blocks;
```

alter database DBS-NAME set 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).

Example:

```
alter database Sample.Basic set implicit_commit after 50 rows;
```

alter database DBS-NAME 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 (&).

Example:

```
alter database Sample.Basic set variable CurrMonth 'Oct';
```

alter database DBS-NAME set default calculation ...

Change the default calculation (which, by default, is `CALC ALL;`) to the stored calculation script you specify, or to an anonymous (unstored) calculation string.

Examples:

```
alter database Sample.Basic set default calculation 'CalcAll';
```

```
alter database Sample.Basic set default calculation as calc_string 'CALC ALL;';
```

alter database DBS-NAME 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.

Example:

```
alter database Sample.Basic set active alias_table 'Long Names';
```

To check which alias table is active, use **query database**.

alter database DBS-NAME set 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](#).

Example:

```
alter database Sample.Basic set performance statistics enabled;
```

alter database DBS-NAME set 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](#).

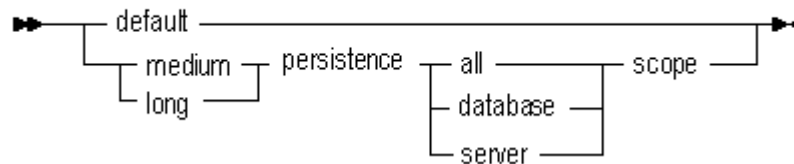
Example:

```
alter database Sample.Basic set performance statistics disabled;
```

alter database DBS-NAME set performance statistics mode to <PST-SPEC>

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> ::=



Example:

```
alter database Sample.Basic set performance statistics mode to default;
```

alter database DBS-NAME set note

Create an informational note about the database that Smart View or other grid client users can see from the login dialog box. The note can be up to 64 kilobytes long.

Example:

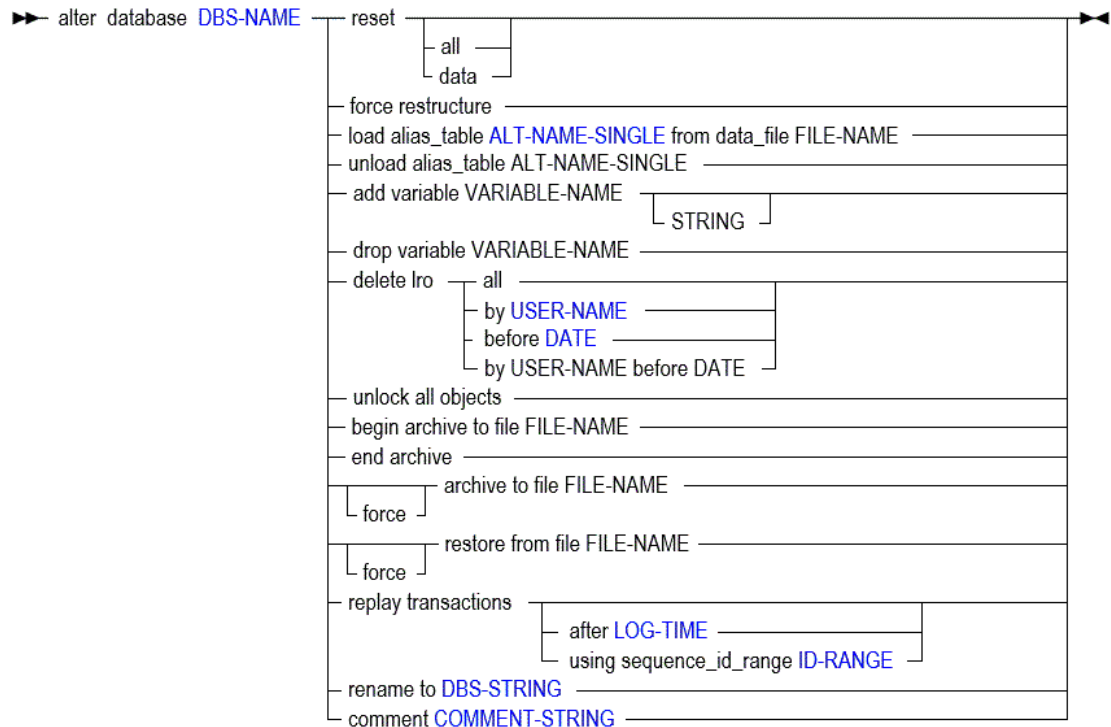
```
alter database Sample.Basic set note 'Calc in progress';
```

Alter Database (Misc)

The MaxL **alter database** statement used with various action keywords helps you perform actions on Essbase databases.

[Click here for aggregate storage version](#)

Syntax



- DBS-NAME
- FILE-NAME
- ALT-NAME-SINGLE
- VARIABLE-NAME
- USER-NAME
- DATE
- LOG-TIME
- ID-RANGE
- DBS-STRING
- COMMENT-STRING

Keywords

Use MaxL **alter database** to change the following database-wide settings. The minimum application permission required for most of the statements is Database Manager, with exceptions noted.

alter database DBS-NAME reset

Clear all data and linked-reporting objects from the database, but preserve the outline.

Example:

```
alter database Sample.Basic reset;
```

alter database DBS-NAME reset all

Clear all data, Linked Reporting Objects, and the outline.

Example:

```
alter database Sample.Basic reset all;
```

alter database DBS-NAME reset data

Clear all data and linked-reporting objects from the database, but preserve the outline (same as using `reset`).

Example:

```
alter database Sample.Basic reset data;
```

alter database DBS-NAME force restructure

Explicitly restructure the database to eliminate or reduce fragmentation. By default, this statement is run in serial. To enable parallel restructuring, use the `RESTRUCTURETHREADS` configuration setting.

Example:

```
alter database Sample.Basic force restructure;
```

alter database DBS-NAME load alias_table ...

Load an alias table from a file to the current database. The feeder file (`FILE-NAME`) must follow these rules:

- Must be correctly formatted.
- Must be located on the Essbase Server, in the cube directory (not on a client computer).
See Environment Locations in the Essbase Platform for information about <Application Directory>, cube directory, and other directory locations in Essbase.
- `FILE-NAME` must include the full path.

Sample contents of a feeder file for loading an alias table:

```
$ALT_NAME
"400-10"      Guava
"400-20"      Tangerine
"400-30"      Mango
$END
```

Example:

```
alter database Sample.Basic load alias_table "Long Names" from data_file
'newalias.alt';
```

Imports the alias table `newalias.alt` into Sample Basic. The file `newalias.alt` must be already uploaded to the Sample Basic cube directory.

alter database DBS-NAME unload alias_table ...

Delete the specified alias table.

Example:

```
alter database Sample.Basic unload alias_table "Long Names";
```

alter database DBS-NAME add variable ...

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 `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.

Example:

```
alter database Sample.Basic add variable Month Oct;
```

alter database DBS-NAME drop variable ...

Remove a substitution variable and its corresponding value from the database.

Example:

```
alter database Sample.Basic drop variable Month;
```

alter database DBS-NAME delete lro ...

Delete Linked Reporting Objects linked to the active database for a given user name or modification date.

Example:

```
alter database Sample.Basic delete lro by user3 before '11/01/2024';
```

alter database DBS-NAME unlock all objects

Unlock all objects on the database that are in use by a user or process. You must be a service administrator. Service administrators can unlock any object, but other users can unlock only those objects that they locked.

Example:

```
alter database Sample.Basic unlock all objects;
```

alter database DBS-NAME begin archive ...

Prepare the database for backup by an archiving program, and prevent writing to the files during backup. You must be a system administrator to perform this action.

This statement requires the database to be started.

Begin archive achieves the following outcomes:

- 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 `end archive`.
- 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.

Begin archive and end archive do not perform the backup; they simply protect the database during the backup process.

 **Note:**

Using the **begin archive to file** and **end archive** grammar is the only supported way to initiate backup and recovery of a database using MaxL.

Example:

```
alter database Sample.Basic begin archive to file 'samplebasic.arc';
```

Backs up the Sample.Basic files to samplebasic.arc.

alter database DBS-NAME end archive

Return the database to read-write mode after backing up the database files.

This statement requires the database to be started.

End archive achieves the following outcomes:

- Returns the database to read-write mode.
- Re-opens database files in exclusive, read-write mode.

 **Note:**

Using the **begin archive to file** and **end archive** grammar is the only supported way to initiate backup and recovery of a database using MaxL.

Example:

```
alter database Sample.Basic end archive;
```

Returns the Sample.Basic database to read-write mode after backing up the database files.

alter database DBS-NAME replay transactions ...

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).

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.

 **Note:**

Transaction logging and replay is available only for backward compatibility support in Essbase 21c. Features added after Essbase 11g On-Premise are not supported for transaction logging and replay; including (but not limited to):

- Batch outline editing
- Application workbooks and Cube Designer activity
- Scenario management
- External data load using a Datasource
- REST API data loads and other updates
- Federated partitions
- MDX inserts
- Drill through transactions

alter database DBS-NAME replay transactions after LOG-TIME

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'

Example:

```
alter database Sample.Basic replay transactions after '11_20_2007:12:20:00';
```

Replays all transactions that were logged after the specified time.

alter database DBS-NAME replay transactions using sequence_id_range ID-RANGE

Replays the transactions specified by a comma-separated list of sequence ID ranges. 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

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.

 **Note:**

You can skip replaying a transaction if you are absolutely sure that the transaction results are not required to recover the database.

Example:

```
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 DBS-NAME rename ...

Rename the database. When you rename a database, the cube directory is also renamed.

Example:

```
alter database Sample.Basic rename to Basic2;
```

alter database DBS-NAME 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 Smart View or other grid client users, use `set note` instead.

Example:

```
alter database Sample.Basic comment 'my comment';
```

Related Topics

- [Alter Database enable | disable](#)
The MaxL **alter database** statement with **enable|disable** keywords helps you turn on and off Essbase database-wide functionality.
- [Alter Database Set](#)
The MaxL **alter database** statement under the **set** keyword helps you modify Essbase database-wide settings.

Alter Drillthrough

The MaxL **alter drillthrough** statement helps you edit drill-through URL definitions linking Essbase to content hosted on Oracle ERP and EPM applications.

Syntax

```

▶▶ alter drillthrough URL-NAME from xml_file FILE-NAME ▶▶
▶ on { MEMBER-EXPRESSION } [ allow_merge ] ▶▶

```

- [URL-NAME](#)
- [FILE-NAME](#)
- [MEMBER-EXPRESSION](#)

Keywords

Use MaxL **alter drillthrough** to edit a URL definition in the following ways:

alter drillthrough

Edit drill-through URL metadata.

The number of drill-through URLs per database is limited to 255.

alter drillthrough URL-NAME 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](#).

alter drillthrough URL-NAME ... 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.

alter drillthrough URL-NAME ... on {<member-expression>,...} allow_merge

Optional: Merge the drillable-region definition instead of replacing it on update.

Example

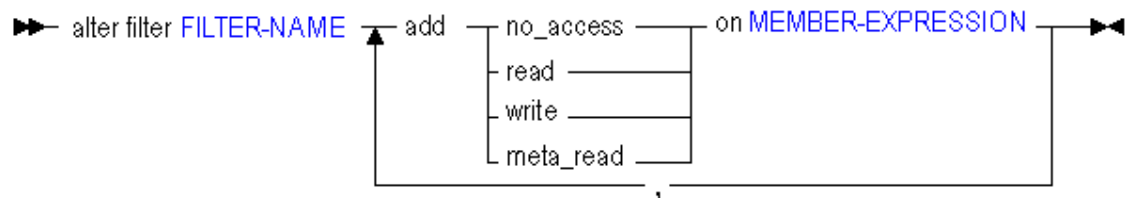
```
alter drillthrough sample.basic.myURL from xml_file "C:/drillthrough/data/
myfile.xml" on {'@Ichildren("Qtr1")', '@Ichildren("Qtr2")'} allow_merge;
```

Alter Filter

The MaxL alter filter statement helps you add rows to an Essbase database's security filter.

Filters control security for database objects. Use [grant](#) to assign filters to users and groups.

Minimum permission required: Database Manager.

Syntax

- **FILTER-NAME**
- **MEMBER-EXPRESSION**

Keywords

Use MaxL **alter filter** to edit security filters in the following ways:

alter filter FILTER-NAME add no_access on <member-expression>

Block access to a specified member combination.

alter filter FILTER-NAME add read on <member-expression>

Provide read-only access to a specified member combination.

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 FILTER-NAME add write on <member-expression>

Provide write access to a specified member combination.

Example:

```
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 filter FILTER-NAME 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](#).

Notes

- Filters created using MaxL must be valid. For information about filter syntax, see [Create Filters](#).
- MEMBER-EXPRESSION must be enclosed in single quotation marks. It can be a comma-separated list.

Alter Group

The MaxL alter group statement helps you remove a filter assignment from an Essbase group.

This statement does not remove filter assignments granted to individual users. To remove filter assignments to users, use [Alter User](#).

Permission required: see [About Filters](#).

Syntax

▶▶ alter group **GROUP-NAME** revoke filter **FILTER-NAME** ◀◀

- **GROUP-NAME**
- **FILTER-NAME**

Example

```
alter group MyGroup revoke filter Sample.Basic.filt1;
```

Alter Object

The MaxL alter object statement helps you rename, unlock, or copy artifacts related to an Essbase database.

Syntax

```

▶▶ alter object OBJ-NAME of type <OBJ-TYPE>
├── rename to OBJ-NAME-SINGLE
├── unlock
├── copy to OBJ-NAME
└── force
▶▶

```

- **OBJ-NAME**
- **OBJ-NAME-SINGLE**
- Object types (OBJ-TYPE) in Essbase 21c that you can work with in MaxL **alter object** are: outline, calc_script, report_file, rules_file, text, partition_file, lro, and worksheet.

Keywords

Use MaxL **alter object** to edit artifacts in the following ways:

alter object OBJ-NAME of OBJ-TYPE rename to

Rename the artifact. Not applicable for partition files, worksheets, or outlines.

Example:

```
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 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 OBJ-NAME of OBJ-TYPE 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 `alter database DBS-NAME unlock all objects;`

Example:

```
alter object samppart.company.company of type partition_file unlock;
```

Unlocks the partition definition file for the Samppart Company database.

alter object OBJ-NAME of OBJ-TYPE 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, the copy fails.

Example:

```
alter object Sample.Basic.'dl_skip' of type rules_file copy to
Sample.Basic.'dl_noskip';
```

alter object OBJ-NAME of OBJ-TYPE force 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. If an administrator issues the statement with the **force** keyword, locked artifacts are unlocked, copied, and re-locked.

Example:

```
alter object Sample.Basic.'dl_skip' of type rules_file force copy to
Sample.Basic.'dl_noskip';
```

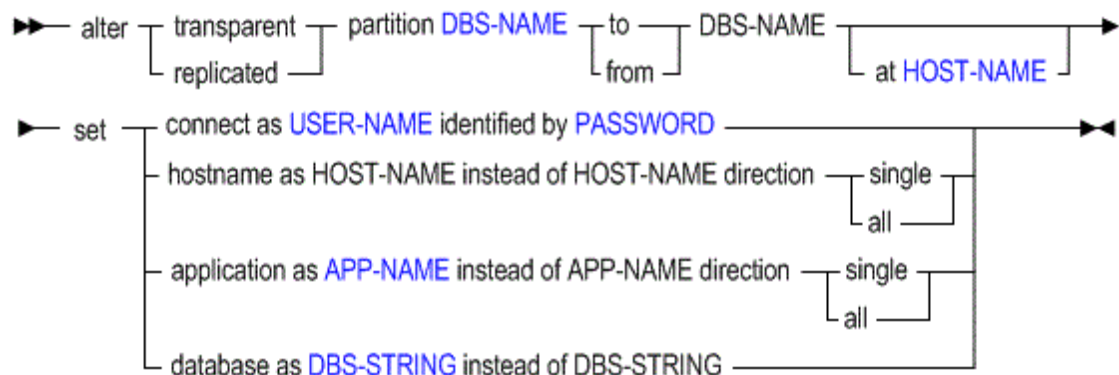
Notes

- Specified artifacts must be persisted in the database directory.
- Attempting to rename or copy an artifact of type `partition_file` returns an error.

Alter Partition

The MaxL alter partition statement helps you fix invalid or dangling Essbase partition references, by changing the authorized user who can connect to both cubes, or by changing application, cube, or host names in the event something was moved or renamed.

Syntax



- DBS-NAME
- HOST-NAME

- `USER-NAME`
- `PASSWORD`
- `APP-NAME`
- `DBS-STRING`

Use MaxL **alter partition** to edit partitions in the following ways:

Keywords

alter...partition...set connect

Change the user authorized to access the partitioned cubes.

alter...partition...set hostname

Edit the partition definition to include the correct URL for the partition source cube, target cube, or both.

alter...partition...set application as

Edit the partition definition to include a corrected application name. This is useful if *one* application name was changed; if both application names changed, the partition definition cannot be corrected and you must re-create it.

alter...partition...set database as

Edit the partition definition to include a corrected cube name. This is useful if *one* cube name was changed; if both cube names changed, the partition definition cannot be corrected and you must re-create it.

alter...partition...direction single

See Examples 2 and 5.

alter...partition...direction all

See Examples 3, 4, and 6.

Notes

- The first DBS-NAME is the local cube, and the second DBS-NAME is the remote cube.
- Directing a partition *to* the remote site means the current cube is the source. Creating a partition *from* the remote site means the current cube is the target.
- To change the authorized partition user, you must change the user for both partitioned cubes, as shown in Example 1.
- If a partitioned host, application, or cube 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. 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. Exception: Host names can be updated on source and target, as shown in Example 4.

Example 1 – Change Partition User

The following example changes the user authorized to access the partitioned cubes.

```
/* To change authorized partition user on target, log in to source & then
use: */
    alter transparent partition appl.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 – Update One Host Name for Target

In the following example, `alter partition` is used to fix a partition definition that became invalid when a URL changed and affected only one half of the partition definition (`app2.target`):

```
alter transparent partition app1.source to app2.target at "https://
myserver1.example.com:9001/essbase/agent"
set hostname as "https://myserver2.example.com:9001/essbase/agent"
instead of "https://myserver1.example.com:9001/essbase/agent" direction
single;
```

where `direction single` indicates that only the target URL needs to be changed.

Example 3 – Update One Host Name for Source and Target

In the following example, `alter partition` is used to fix a partition definition that became invalid when one 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 "https://
myserver2.example.com:9001/essbase/agent"
set hostname as "https://myserver2.example.com:9001/essbase/agent"
instead of "https://myserver1.example.com:9001/essbase/agent" 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 – Update Host Name for Source and Target

In the following example, `alter partition` is used to fix a partition definition that became invalid when target and source cubes that were on different hosts are both moved to the same new host:

```
alter transparent partition app1.target from app2.source set hostname as
"https://myserver3.example.com:9001/essbase/agent" instead of "https://
myserver1.example.com:9001/essbase/agent" direction all;
```

OR

```
alter transparent partition app1.target from app2.source set hostname as
"https://myserver3.example.com:9001/essbase/agent" instead of "https://
myserver2.example.com:9001/essbase/agent" 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 5 – Update Source Application Name

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 6 – Update Application Name for Source and Target

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 cubes were on the same application:

```
alter transparent partition newAppName.source to newAppName.target
    set application as newAppName instead of oldAppName direction all;
```

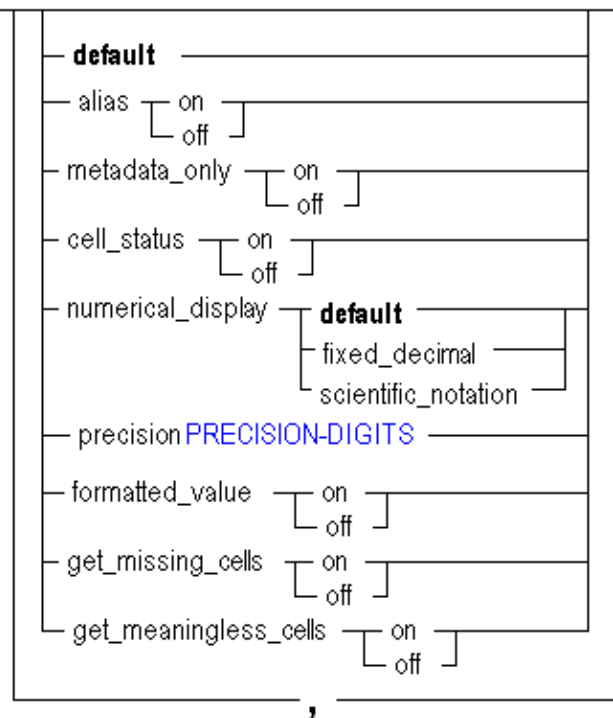
where `direction single` indicates both halves of the partition definition need to be corrected.

Alter Session

The MaxL `alter session` statement helps you set MDX display options for Essbase.

Syntax

alter session set dml_output



PRECISION-DIGITS

Use MaxL **alter session** to change the following MDX output settings:

Keywords

alter session set dml_output default

Revert to the default MDX display settings in the MaxL Shell. The default settings are: alias ON, metadata_only OFF, cell_status OFF.

Example:

```
alter session set dml_output default;
```

alter session set dml_output alias on|off

Set whether to use aliases instead of member names.

Example:

```
alter session set dml_output alias off;
```

alter session set dml_output metadata_only on|off

Set whether to show only the metadata, with no data.

Example:

```
alter session set dml_output metadata_only on;
```

alter session set dml_output cell_status on|off

Set whether to display cell status. Cell status is additional information returned with each cell value in MDX query outputs.

Example:

```
alter session set dml_output cell_status off;
```

 **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.

alter session set dml_output 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).

Example:

```
alter session set dml_output numerical_display fixed_decimal;
```

alter session set dml_output precision <precision-digits>

Set the number (0-15) of decimal places to include for the data values in MDX query output.

Example:

```
alter session set dml_output precision 3;
```

alter session set dml_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.

Example:

```
alter session set dml_output formatted_value off;
```

alter session set dml_output 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.

Example:

```
alter session set dml_output get_missing_cells off;
```


alter session set dml_output get_meaningless_cells on|off

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.

The following example query gets sales for all products, but the aggregation is specified by the slicer context only for Ounces_12.

```
SELECT
{Sales, Cogs}
ON COLUMNS,
  {Product.Levels(0).Members}
ON ROWS
FROM Sample.Basic
WHERE (Ounces_12)
;
```

A value of #Meaningless is displayed for any members not associated with the attribute Ounces_12.

Example:

```
alter session set dml_output get_meaningless_cells off;
```

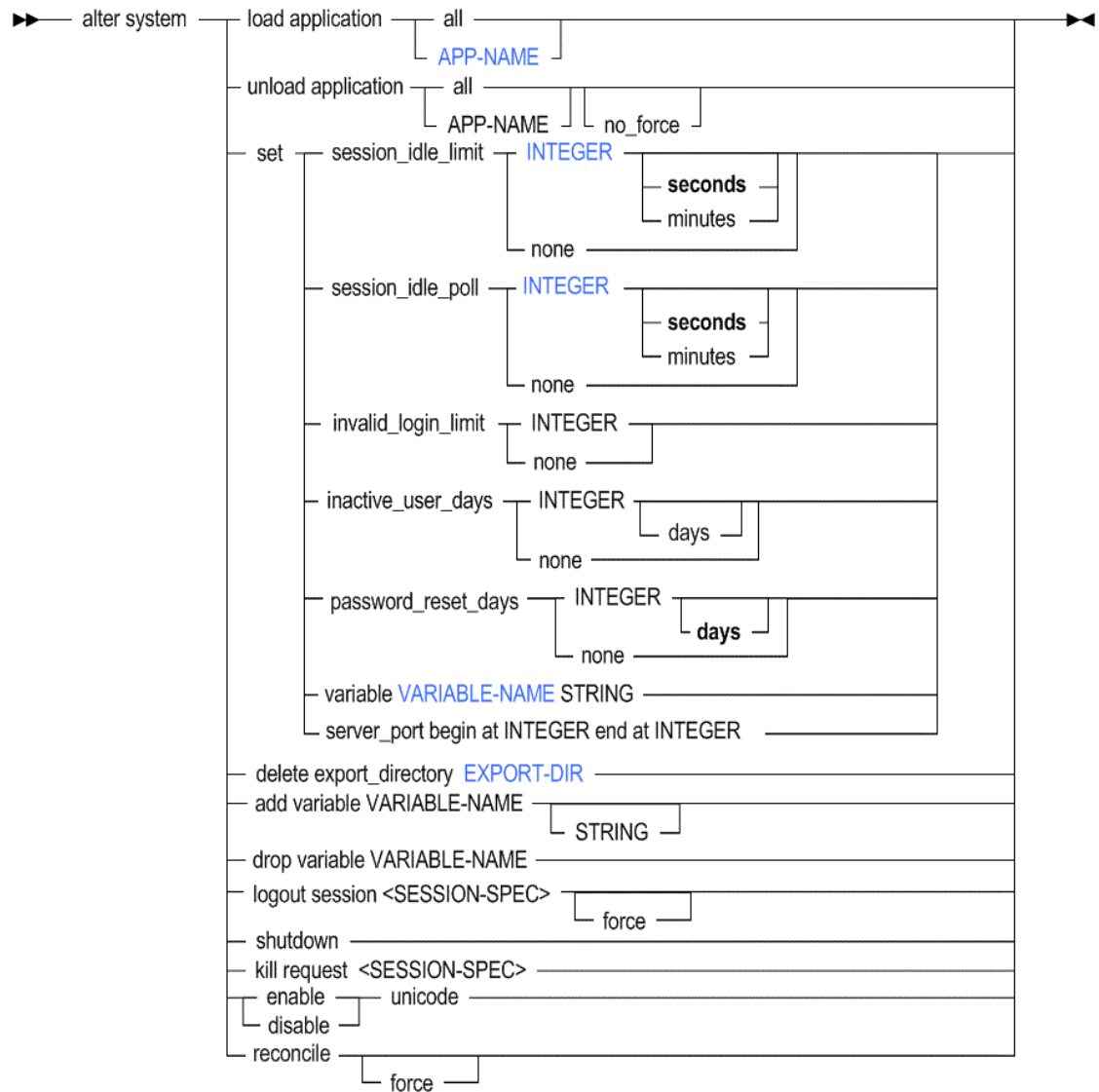
Alter System

The MaxL alter system statement helps you change the state of the Essbase Server.

[Click here for aggregate storage version](#)

You can use this statement to start and stop applications, change system-wide variables, manage password and login activity, disconnect users, end processes, or shut down the server.

Syntax



- APP-NAME
- INTEGER
- VARIABLE-NAME
- EXPORT-DIR

Keywords

Use MaxL **alter system** to change the following Essbase Server-wide settings. The role required for most of the statements is System Administrator, with exceptions noted.

alter system load application

Start an application, or start all applications on the Essbase Server. Requires at least Database Access permission for the application.

Example:

```
alter system load application Sample;
```

alter system unload application

Stop an application, or stop all applications on the Essbase Server. Requires at least Database Access permission for the application. Unloading an application cancels all active requests and database connections, and stops the application. If Essbase encounters a problem when trying to cancel active requests and database connections, and stopping the application, an error is logged in the application log.

If you do not want to stop an application if it has active requests and database connections, use the **no_force** keyword. When using **no_force**:

- If the application has active requests and database connections, the application is not stopped; it continues running and returns an error
- If the application does not have active requests and database connections, the application is stopped, as if you used **unload application** without specifying **no_force**

Examples:

```
alter system unload application Sample;
```

Stops the Sample application, if it is currently running.

```
alter system unload application all;
```

Terminates all active requests and stops all applications.

```
alter system unload application Sample no_force;
```

Essbase prepares to unload the Sample application; however, if active requests are running, the application is not stopped.

alter system set session_idle_limit ...

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.

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.

Because users may mistakenly assume that the calculation stopped when they were logged out, you can do one of the following to correct the user experience:

- 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

Example:

```
alter system set session_idle_limit 60 minutes;
```

alter system set session_idle_poll ...

Set the time interval for inactivity checking. The time interval specified in the session idle poll tells Essbase how often to check whether user sessions have passed the allowed inactivity interval indicated by `session_idle_limit`.

Example:

```
alter system set session_idle_poll 300 seconds;
```

alter system set invalid_login_limit ...

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.

Example:

```
alter system set invalid_login_limit 3;
```

alter system set inactive_user_days ...

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.

Example:

```
alter system set inactive_user_days 30;
```

alter system set password_reset_days ...

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.

Example:

```
alter system set password_reset_days 60;
```

Specifies that all users will be prompted after 60 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 variable ...

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 (&).

Example:

```
alter system set variable Month Nov;
```

alter system set server_port ...

Expand the port range specified by Essbase configuration properties. Each Essbase application uses two ports from this range. If no more ports are available, an error message is displayed.

 **Note:**

You can expand port ranges only so that the beginning port range is less than SERVERPORTBEGIN and the ending port range is greater than SERVERPORTEND.

Example:

```
alter system set server_port begin at 32750 end at 33780;
```

alter system delete export_directory ...

Delete directories created for linked reporting objects exported from a block storage (BSO) database to a directory created in the application directory. Use this grammar after the exported LROs are migrated into a database using **import lro**, and the directories containing the exported LRO information are not needed.

 **Note:**

This process works only for directories created in the application directory, using the DBS-EXPORT-DIR option of the **export lro** statement. It does not work for directories created elsewhere using the FULL-EXPORT-DIR option of the **export lro** statement.

To view a list of names of exported linked-reporting-objects directories in the application directory, use `display system export_directory`.

Example:

```
alter system delete export_directory Sample-Basic-export;
```

alter system 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.

Example:

```
alter system add variable DSN 'TBC_DB2';
```

alter system drop variable

Remove a substitution variable and its corresponding value from the system.

Example:

```
alter system drop variable DSN;
```

alter system logout session all

Terminate all user sessions running on the Essbase Server.

Example:

```
alter system logout session all;
```

alter system logout session...force

Terminate a session (or sessions) even if it is processing a request. The request is allowed to proceed to a safe point, and then the transaction is rolled back.

Example:

```
alter system logout session 3694133190 force;
```

alter system logout session SESSION-ID

Terminate a session by its unique session ID number. To see the session ID number, use [display session](#).

Example:

```
alter system logout session 3694133190;
```

alter system logout session by user USER-NAME

Terminate all current sessions by a particular user, either across the entire Essbase Server, or limited to a specific application or database.

Example:

```
alter system logout session by user User1;
```

Disconnects User1 from any applications or databases to which they are connected.
To log out a user, log out the sessions owned by that user.

alter system logout session by user ... on application ...

Terminate all current sessions by a particular user across a specific application.

Example:

```
alter system logout session by user User1 on application Sample;
```

alter system logout session by user ... on database ...

Terminate all current sessions by a particular user across a specific database.

Example:

```
alter system logout session by user User1 on database Sample.Basic;
```

alter system logout session on application ...

Terminate all current user sessions across a specific application.

Example:

```
alter system logout session on application Sample;
```

alter system logout session on database ...

Terminate all current user sessions across a specific database.

Example:

```
alter system logout session on database Sample.Basic;
```

alter system shutdown

Shut down the Essbase Server.

Example:

```
alter system shutdown;
```

Stops all running applications and shuts down Essbase Server.

alter system kill request all

Terminate all current requests on the Essbase Server.

Note:

To terminate your own active request in MaxL Shell, press the ESC key.

Example:

```
alter system kill request all;
```

alter system kill request SESSION-ID

Terminate the current request indicated by the session ID. You can obtain session IDs using [display session](#).

Example:

```
alter system kill request 3694133190;
```

alter system kill request by user ...

Terminate all current requests by the specified user on the Essbase Server.

Example:

```
alter system kill request by user User1;
```

alter system kill request on application ...

Terminate all current requests on the specified application.

Example:

```
alter system kill request on application Sample;
```

alter system kill request on database ...

Terminate all current requests on the specified database.

Example:

```
alter system kill request on database Sample.Basic;
```

alter system 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.

Example:

```
alter system enable unicode;
```

alter system 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.

Example:

```
alter system disable unicode;
```

alter system reconcile

No longer applicable.

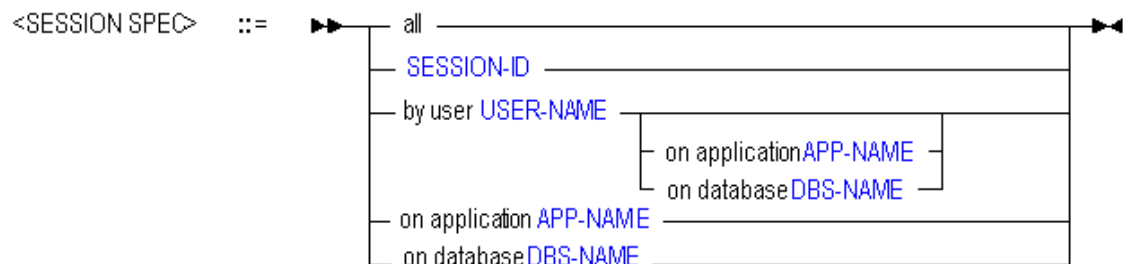
Notes

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.



- SESSION-ID
- USER-NAME
- APP-NAME
- DBS-NAME

Alter Trigger

The MaxL alter trigger statement helps you enable or disable a trigger to track state changes over a selected Essbase cube area.

Syntax



- TRIGGER-NAME
- DBS-NAME

Use MaxL **alter trigger** to edit triggers in the following ways:

Keywords

alter trigger TRIGGER-NAME enable

Essbase monitors the trigger during data load, calculation, or lock and send, and performs the trigger action when the condition is met on the specified cube area.

Example:

```
alter trigger Sample.Basic.WatchCosts enable;
```

alter trigger TRIGGER-NAME disable

Essbase does not monitor the trigger.

Example:

```
alter trigger Sample.Basic.WatchCosts disable;
```

alter trigger on database <DBS-NAME> disable

Disables all triggers 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 `alter trigger on database DBS-NAME disable` are re-enabled).

Example:

```
alter trigger on database sample.basic disable;
```

Alter User

The MaxL alter user statement helps you remove a filter assignment from an Essbase user.

This statement does not remove filter assignments gained by membership to groups. To remove filter assignments to groups, use [Alter Group](#).

Permission required: see About Filters.

Syntax

▶ alter user **USER-NAME** revoke filter **FILTER-NAME** ▶▶

- **USER-NAME**
- **FILTER-NAME**

Example

```
alter user Fiona revoke filter Sample.basic.filt1;
```

Create Application

The MaxL **create application** statement for block storage (BSO) helps you create or re-create an Essbase application. Every Essbase database (cube) must be created within an application. If the cube is intended to be block storage, then the application must be block storage.

[Click here for aggregate storage version](#)

You can create the application either from scratch or as a copy of another application on the same server. The permission required to create an application is power user or system administrator.

See **APP-NAME** for information on the maximum length of and special characters that are allowed in an application name. Application names are not case-sensitive.

Syntax

```
▶▶ create [ or replace ] application APP-NAME [ type [ nonunicode_mode | unicode_mode ] ]
[ as APP-NAME ] [ comment COMMENT-STRING ] ▶▶
```

- **APP-NAME**
- **COMMENT-STRING**

Keywords

Use the MaxL **create application** statement to create a block storage (BSO) application in the following ways.

create application APP-NAME

Create a new application intended for a block storage (BSO) cube.

Example:

```
create application MyBSOApp;
```

Creates an application called MyBSOApp.

create or replace application ...

Create an application intended for a BSO cube, replacing an existing application of the same name, if one exists.

Example:

```
create or replace application MyBSOApp;
```

Creates an application called MyBSOApp. If an application named MyBSOApp already exists, it is overwritten.

create application APP-NAME type nonunicode_mode

Create a non Unicode-mode application intended for a block storage database. Non-Unicode mode is the default application type that is created, even if these keywords are omitted.

Example:

```
create application MyBSOApp type nonunicode_mode;
```

create application APP-NAME type unicode_mode

Create a Unicode mode application intended for a block storage database.

Example:

```
create application MyBSOApp type unicode_mode;
```

create application APP-NAME as APP-NAME

Create an application as a copy of another application. To copy an application, Application Manager permission on the source application is required, in addition to the power user role required to create an application.

Example:

```
create application MyBSOApp as Sample;
```

Creates a block storage application called MyBSOApp, which is a copy of the gallery sample application named Sample.

create application APP-NAME comment COMMENT-STRING

Create a block storage application including a short description (optional). The description can contain up to 80 characters.

Example:

```
create application MyBSOApp comment 'This is a test application.';
```

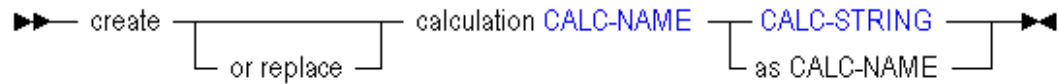
Create Calculation

The MaxL create calculation statement helps you create, replace, or copy an Essbase block storage (BSO) calculation script.

The minimum application permissions required to create calculation scripts are:

- Database Manager to create database-level calculations.
- Application Manager to create application-level calculations.

Syntax



- [CALC-NAME](#)
- [CALC-STRING](#)

Keywords

You can use the MaxL **create calculation** statement to create a BSO calculation in the following ways:

create calculation CALC-NAME CALC-STRING

Create a calculation script, the body of which is specified by [CALC-STRING](#).

create or replace calculation CALC-NAME CALC-STRING

Create a calculation script, the body of which is specified by [CALC-STRING](#). If a calculation script of that name already exists, it is replaced.

Example:

```

create or replace calculation sample.basic.Accts
'SET UPDATECALC ON;
CALC DIM(Accounts);'
;

```

Creates a calculation named Accts that is associated with the Sample Basic database.

create calculation CALC-NAME as CALC-NAME

Create a block storage calculation script as a copy of an existing calculation.

Example:

```

create calculation Sample.Accts2 as Sample.Basic.Accts

```

Creates a calculation named Accts2 on the application named Sample. Accts2 is a copy of the Sample Basic database-level calculation named Accts.

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](#) for more details.
- Calculations created using MaxL must be valid. For information about calculation syntax, see Understanding Calculation Script Syntax.

Create Database

The MaxL create database statement helps you create or re-create an Essbase cube, optionally as a copy of another on the same server.

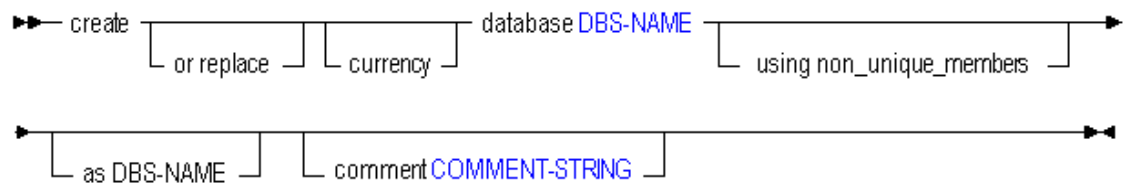
[Click here for aggregate storage version](#)

See **DBS-NAME** for information on the maximum length of and special characters that are allowed in a database name. Database names are not case-sensitive.

The syntax for creating a block storage database is the same as for creating an aggregate storage database, but you must create the block storage database within an existing block storage application.

The minimum permission required to create a database is Application Manager. To copy a database, at least Database Manager permission on the source database is also required.

Syntax



- **DBS-NAME**
- **COMMENT-STRING**

Keywords

Use the MaxL **create database** statement to create an Essbase cube in the following ways:

create database DBS-NAME

Create a new database.

Example:

```
create database MyBSOApp.Basic;
```

create or replace database DBS-NAME

Create a database, or replace an existing database of the same name.

Example:

```
create or replace database MyBSOApp.Basic;
```

create database DBS-NAME 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.

Example:

```
create database MyBSOApp.Basic using non_unique_members;
```

For more information about duplicate member names, see [Creating and Working With Duplicate Member Outlines](#).

create database DBS-NAME as ...

Create a database as a copy of another database.

Example:

```
create database MyBSOApp.New as Sample.Basic;
```

Creates a database called New within the MyBSOApp application that is a copy of the database Basic within the Sample application.

create currency database DBS-NAME

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.

Example:

```
create currency database Sample.Intern1;
```

Creates a currency database called Intern1 within the Sample application.

create database DBS-NAME comment

Create a database description (optional). The description can contain up to 80 characters.

Example:

```
create or replace database MyBSOApp.Basic comment 'Test cube.';
```

Creates a database called Basic within the MyBSOApp application, with a descriptive comment.

Create Drillthrough

The MaxL create drillthrough statement helps you create a drill-through URL within the active Essbase cube outline. For each drillable region, you can enable URL access to Web content hosted on Oracle ERP and EPM applications. The limit of drill-through URLs per cube is 255.

Syntax

```

▶▶ create drillthrough URL-NAME from xml_file FILE-NAME ▶▶
▶ on — { MEMBER-EXPRESSION } [level0 only] ▶▶

```

- [URL-NAME](#)

- [FILE-NAME](#)
- [MEMBER-EXPRESSION](#)

Use the MaxL **create drillthrough** statement to create an Essbase drill-through URL definition in the following ways:

Keywords

create drillthrough URL-NAME from xml_file FILE-NAME

The **from xml_file** clause indicates 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 cube. The XML contains the drill-through URL display name and a URL enabling the hyperlink from a cell to a Web interface to occur.

The following is a sample URL XML file:

```
<?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>
```

create drillthrough ... on {<member-expression>,...}

The **on MEMBER-EXPRESSION** clause defines the list of drillable regions, using the same 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.

create drillthrough ... level0 only

The optional **level0 only** clause restricts the URL definition to level-0 data.

Example

```
create drillthrough sample.basic.myURL from xml_file "myfile1.xml" on
{'@Ichildren("Qtr1")', '@Ichildren("Qtr2")'} level0 only;
```

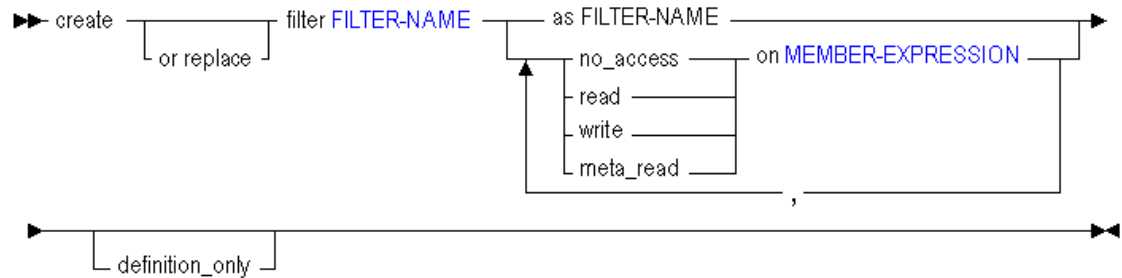
Create Filter

The MaxL **create filter** statement helps you create or re-create an Essbase security filter, either from scratch or as a copy of another filter on the same server.

Filters control security for Essbase database objects. Use [grant](#) to assign filters to users and groups.

Minimum permission required: Database Manager.

Syntax



- **FILTER-NAME**
- **MEMBER-EXPRESSION**

Use the MaxL **create filter** statement to create a filter in the following ways:

Keywords

create filter FILTER-NAME

Create a security filter to restrict or permit access to specified database cells.

create or replace filter FILTER-NAME

Create a security filter or replace an existing security filter of the same name.

create filter FILTER-NAME no_access on <member-expression>

Create a filter blocking access to a specified member combination.

create filter FILTER-NAME read on <member-expression>

Create a filter providing read-only access to a specified member combination.

create filter FILTER-NAME write on <member-expression>

Create a filter providing write access to a specified member combination.

create filter FILTER-NAME 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](#).

create or replace filter FILTER-NAME ... definition_only;

Updates the filter definition while retaining user associations with the filter. If you replace a filter without using `definition_only`, then the filter must be re-granted to any users to whom it was assigned.

Notes

The 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

The MaxL create function statement helps you create and register your custom-defined Essbase calculation function (CDF), using a Java method.

Minimum permission required:

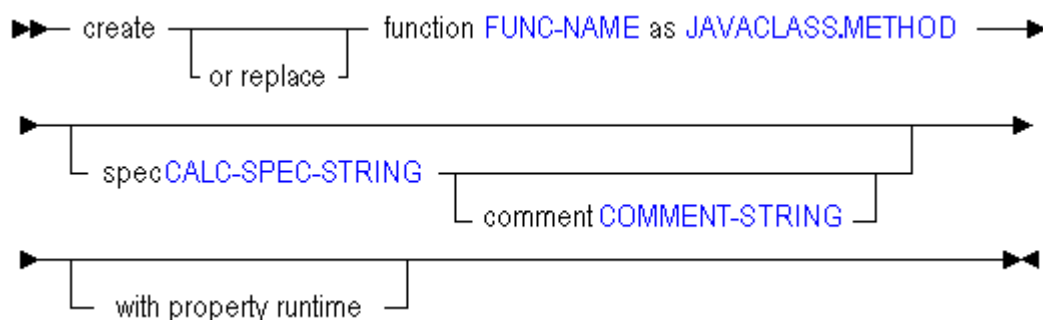
- Application Manager to create a local (application-level) function.
- Administrator to create a global (system-level) function.

The general workflow for creating your own custom-defined functions in Essbase is as follows:

1. Develop the functions in Java classes.
2. Use the **create function** MaxL statement to register your functions in the Essbase calculator framework.
3. You can now use the functions in the same way that you use the standard Essbase calculation functions.

For full information, refer to Developing Custom-Defined Calculation Functions.

Syntax



- **FUNC-NAME**
- **JAVACLASS.METHOD**
- **CALC-SPEC-STRING**
- **COMMENT-STRING**

Use the MaxL **create function** statement to create a function in the following ways:

Keywords

create function FUNC-NAME as JAVACLASS.METHOD

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**. To register a local (application-wide) function, use two tokens for **FUNC-NAME**.

create or replace function FUNC-NAME as JAVACLASS.METHOD

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.

create or replace function ... spec

Enter, for the custom-defined function, an optional Essbase calculator-syntax specification string, such as in the following example: `@COVARIANCE (expList1, expList2)`. Use a specification string if you wish the function to be returned by the output string of the `IEssCube.getCalcFunctions` Java method or `EssListCalcFunctions` C API function.

 **Note:**

If you do not specify a calculation specification string, you cannot specify a comment either.

create or replace function ... with property runtime

The optional **with property runtime** clause designates the custom-defined function as a runtime function. Normally, Essbase pre-executes CDFs 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 CDF 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 CDF 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 CDFs that are applied to constants.

create or replace function ... spec CALC-SPEC-STRING comment

The optional **comment** clause enables you to include a description of the custom-defined function. 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 `IEssCube.getCalcFunctions` Java method or `EssListCalcFunctions` C 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

The following example registers a global CDF named @COVARIANCE. The code for this sample function is available on the Essbase Server, in <Oracle Home>/essbase/products/Essbase/EssbaseServer/java/essbase.jar.

```
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 an Essbase security group using MaxL. This statement is supported for use in EPM Shared Services security mode only. You can only create groups using **type external** grammar, to provision groups that exist in EPM Shared Services.

Syntax

```
➤ create group GROUP-NAME type external ➤
```

GROUP-NAME

Use the MaxL **create group** to create an Essbase security group in the following ways:

Keywords

create group GROUP-NAME type external

Create a security group to assign users to, so that they can share identical minimum permissions assigned at the group level.

For use in EPM Shared Services security mode only. Create and provision in Essbase a group that already exists in EPM Shared Services.

Example

```
create group Level_1 type external;
```

Create Location Alias

The MaxL create location alias statement helps you create on the Essbase database a location alias referencing another cube, to support cross-cube calculations. Location aliases are used in conjunction with @XREF and @XREF calculation functions.

Minimum permission required: Database Manager.

Syntax

```

create [ or replace ] location alias [ LOC-ALIAS-SINGLE from DBS-NAME
                                         LOCATION-ALIAS-NAME ]
to DBS-NAME at HOST-NAME as USER-NAME identified by PASSWORD

```

- LOC-ALIAS-SINGLE
- LOCATION-ALIAS-NAME
- DBS-NAME
- HOST-NAME
- USER-NAME
- PASSWORD

Use the MaxL **create location alias** statement to create a location alias in the following ways:

Keywords

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 or @XWRITE functions 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.

create or replace location alias ... from <db-name>

Specify the name of the current database (the database on which the location alias is being created).

create or replace location alias ... to <db-name>

Specify the name of the remote database to log in to.

create or replace location alias ...at <host-name>

Specify where the remote database resides (using discovery URL).

create or replace location alias ...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.
- Location aliases are used by the @XREF and @XWRITE calculation functions for cross-database calculations.

Example

```

create location alias EasternDB from Sample.Basic to East.Sales at "https://
192.0.2.1:443/essbase/agent" as adminuser identified by 'password';

```

Creates a location alias called EasternDB on Sample.Basic that represents the following login information:

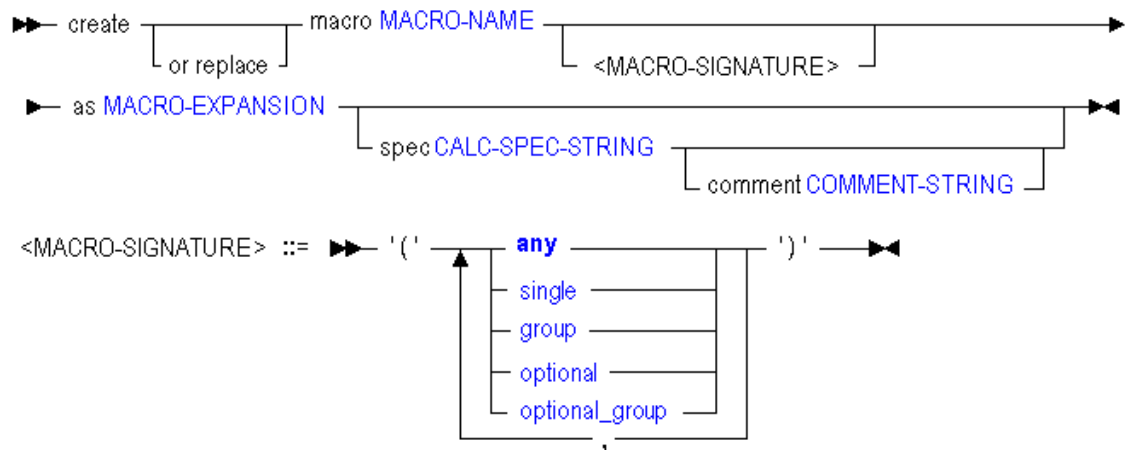
- remote host = https://192.0.2.1:443/essbase/agent
- application = East
- database = Sales
- user name = adminuser
- password = password

Create Macro

The MaxL create macro statement helps you create and register 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 is Application Manager to create a local (application-level) macro, or system administrator to create a global (system-level) macro.

Syntax



- **MACRO-NAME**
- **MACRO-EXPANSION**
- **CALC-SPEC-STRING**
- **COMMENT-STRING**
- **MACRO-SIGNATURE**

Use the MaxL **create macro** statement to create a macro in the following ways:

Keywords

create macro MACRO-NAME as MACRO-EXPANSION

Create and register 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**. To register a local (application-wide) function, use two tokens.

create macro MACRO-NAME <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. See Custom-Defined Macro Input Parameters.

create or replace macro MACRO-NAME ...

Register 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.

create macro MACRO-NAME as MACRO-EXPANSION spec CALC-SPEC-STRING

The optional **spec** clause is a 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 IEssCube.getCalcFunctions Java method or EssListCalcFunctions C API function.



Note:

If you do not specify a calculation specification string, you cannot specify a comment either.

create macro MACRO-NAME as MACRO-EXPANSION spec CALC-SPEC-STRING comment COMMENT-STRING

The optional **comment** clause enables you to add a description of the macro (optional). You cannot create a comment without also using **spec** clause to create a calculator-syntax specification string. The optional calculator-syntax specification string and the comment are used as the output string of the IEssCube.getCalcFunctions Java method or EssListCalcFunctions C 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

The MaxL create partition statement helps you create or validate a partition definition between two Essbase databases.

Permission required: Database Manager at both sites.

Select the type of partition to create:

- transparent
- replicated

To create a federated partition, do not use MaxL. Instead, see Integrate Essbase with Autonomous Database Using Federated Partitions.

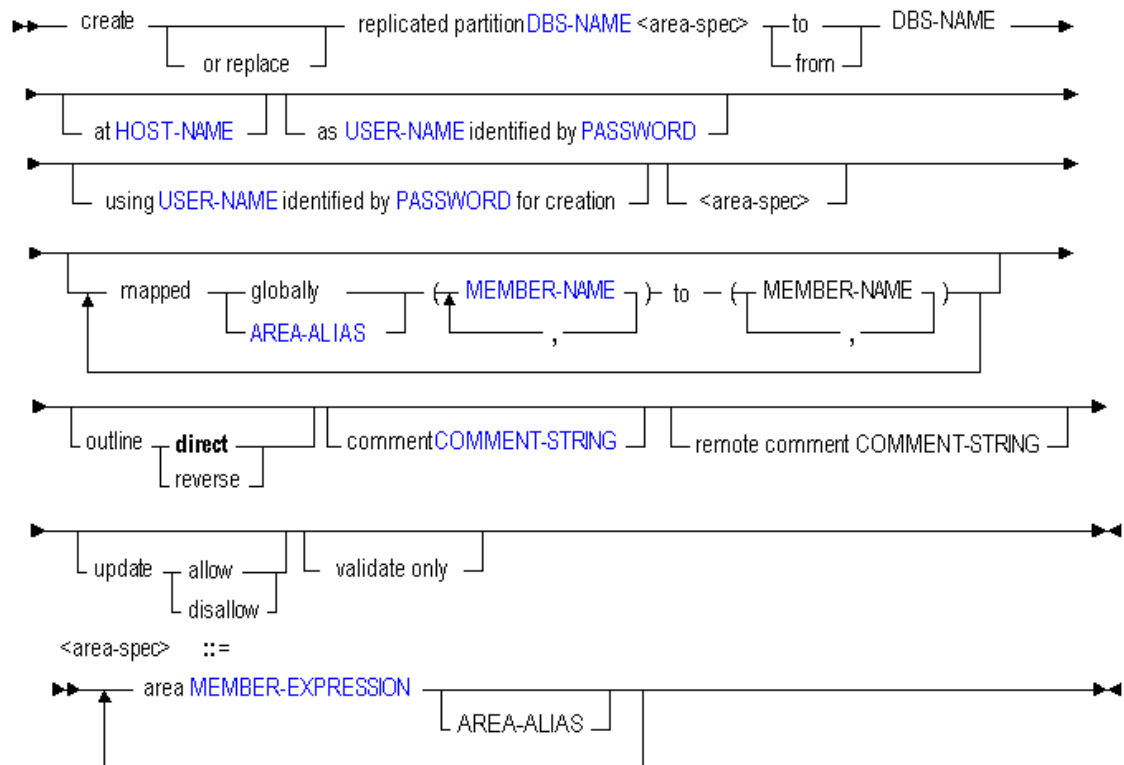
Partitions created using MaxL must be valid. To validate a partition, use the **validate only** clause.

Create Replicated Partition

The MaxL create replicated partition statement helps you create or validate a replicated partition between two Essbase databases.

A replicated partition copies a portion of the source (or originating) cube to be stored in a target cube. Users can access the target cube as if it were the source. The administrator must periodically **refresh** the target data from the source data.

Syntax



- DBS-NAME
- HOST-NAME
- USER-NAME
- PASSWORD
- AREA-ALIAS
- MEMBER-NAME
- COMMENT-STRING

- **MEMBER-EXPRESSION**

Keywords

Use the MaxL **create replicated partition** statement to create a partition in the following ways:

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 replicated partition...

Create a partition definition, or replace an existing partition definition.

create replicated partition ... area ...

The area specification is where you define the partition areas to share with the other cube. Optionally, nickname the area using an [area-alias](#).

create replicated partition ... to DBS-NAME

Use the **to** clause to create a partition definition between the current source cube and the second cube (the target).

create replicated partition ... from DBS-NAME

Use the **from** clause to create a partition definition between the current target cube and the second cube (the source).

create replicated partition ... at HOST-NAME

Use the **at** clause, with a host name, to specify the discovery URL (ending in **/agent**) of the remote cube.

create replicated partition ... as USER-NAME identified by PASSWORD

Use the **as** clause to provide the name and password of a default partition user who can connect to both cubes. Essbase uses the login information to:

- Transfer data between the source and the target for replicated and transparent partitions. Security filters can be applied to prevent end users from seeing privileged data.
- Synchronize outlines for all partition types.

create replicated partition ... using USER-NAME identified by PASSWORD for creation

Use the **using** clause to 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.

create replicated partition ... mapped...

Use the **mapped** clause to define the member-name mapping for shared sections of both cubes, if member names for sections that map are different in the two cubes.

create replicated partition ... outline...

Use the **outline** clause to specify the direction in which outline synchronization should proceed, if necessary. The default direction is the same as the data-refresh direction.

create replicated partition ... update...

Use the **update** clause to 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.

create replicated partition ... comment

Use the **comment** clause to create a comment to describe the source half of the partition definition.

create replicated partition ... remote comment

Use the **remote comment** clause to create a comment to describe the target half of the partition definition.

create replicated partition ... validate only

Use the **validate only** clause to validate the existing partition definition described by this statement, without actually creating it.

Notes

- Multiple area specifications are allowed, provided they are separated by space. Multiple mappings are allowed, provided they are separated by space. 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 cube, and the second DBS-NAME is the remote cube.
- Creating a partition *to* the remote site means the current cube is the source. Creating a partition *from* the remote site means the current cube is the target.
- Aggregate storage cubes can be the target, but not the source, of a replicated partition.

Example

```
create or replace replicated partition source.source
area 'DimensionA' sourceAreaA
area 'DimensionB' sourceAreaB
to target.target at "https://myserver.example.com:9001/essbase/agent"
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 cube Source to cube 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 "https://myserver.example.com:9001/essbase/agent"
as partitionuser identified by 'password'
area '@IDESCENDANTS(East) @IDESCENDANTS(Qtr1)'
update disallow;
```

Creates a replicated partition from an area in the source cube, sampeast.east, to an area in the target cube, samppart.company.

```
create or replace replicated partition sampeast.east
area '@IDESCENDANTS("Eastern Region"), @IDESCENDANTS(Qtr1) '
to samppart.company at "https://myserver.example.com:9001/essbase/agent"
as admin identified by 'password'
area '@IDESCENDANTS(East) @IDESCENDANTS(Qtr1) ' myalias
mapped myalias (Year) to (Yr)
update allow validate only;
```

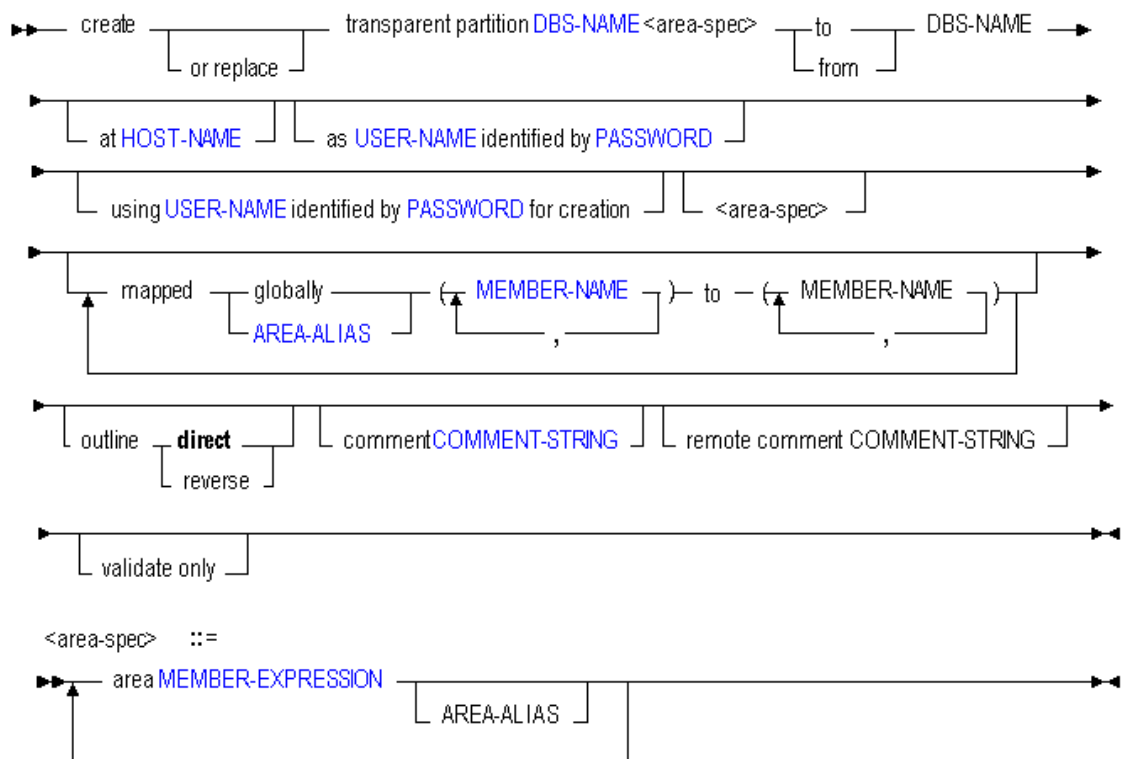
Validates the syntax of a replicated partition you might want to create. To create the partition after checking validity, simply remove the *validate only* phrase. For an explanation of *myalias* as used above, see the definition for [AREA-ALIAS](#).

Create Transparent Partition

The MaxL create transparent partition statement helps you create or validate a transparent partition between two Essbase databases.

A transparent partition allows users to manipulate data that is stored in a target cube as if it were part of the source cube. The remote data is retrieved from the data source each time that users at the data target request it.

Syntax



- DBS-NAME
- HOST-NAME
- USER-NAME
- PASSWORD
- AREA-ALIAS
- MEMBER-NAME
- COMMENT-STRING
- MEMBER-EXPRESSION

Keywords

Use the MaxL **create transparent partition** statement to create a partition in the following ways:

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 cube, or on another Essbase instance.

create or replace transparent partition ...

Create a partition definition, or replace an existing partition definition.

create transparent partition ... area...

The area specification is where you define the partition areas to share with the other cube. Optionally, nickname the area using an [area-alias](#).

create transparent partition ... to DBS-NAME

Use the **to** clause to create a partition definition between the current source cube and the second cube (the target).

create transparent partition ... from DBS-NAME

Use the **from** clause to create a partition definition between the current target cube and the second cube (the source).

create transparent partition ... at HOST-NAME

Use the **at** clause, with a host name, to specify the discovery URL (ending in **/agent**) of the remote cube.

create transparent partition ... as USER-NAME identified by PASSWORD

Use the **as** clause to provide the name and password of a default partition user who can connect to both cubes. Essbase uses the login information to:

- Transfer data between the source and the target for replicated and transparent partitions. Security filters can be applied to prevent end users from seeing privileged data.
- Synchronize outlines for all partition types.

create transparent partition ... using USER-NAME identified by PASSWORD for creation

Use the **using** clause to 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.

create transparent partition ... mapped...

Use the **mapped** clause to define the member-name mapping for shared sections of both cubes, if member names for sections that map are different in the two cubes.

create transparent partition ... outline...

Use the **outline** clause to specify the direction in which outline synchronization should proceed, if necessary. The default direction is the same as the data-refresh direction.

create transparent partition ... comment

Use the **comment** clause to create a comment to describe the source half of the partition definition.

create transparent partition ... remote comment

Use the **remote comment** clause to create a comment to describe the target half of the partition definition.

create transparent partition ... validate only

Use the **validate only** clause to 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 cube, and the second DBS-NAME is the remote cube.
- Creating a partition *to* the remote site means the current cube is the source. Creating a partition *from* the remote site means the current cube is the target.
- Aggregate storage cubes 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 cubes.

Example

```
create or replace transparent partition sampeast.east
    area '@CHILDREN("Eastern Region"), @CHILDREN(Qtr1)' sourceArea
to samppart.company at "https://myserver.example.com:9001/essbase/agent"
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 "https://myserver.example.com:9001/essbase/agent"
as smith 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 cube Source to cube 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

The MaxL create trigger statement helps you create or replace a trigger to track state changes over a selected Essbase cube area.

Select the type of trigger to create:

- [on-update](#)
- [after-update](#)

Create After-Update Trigger

The MaxL create after update trigger statement helps you create or replace a trigger to track state changes over a selected Essbase cube area. The trigger is activated after a data update operation completes.

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.

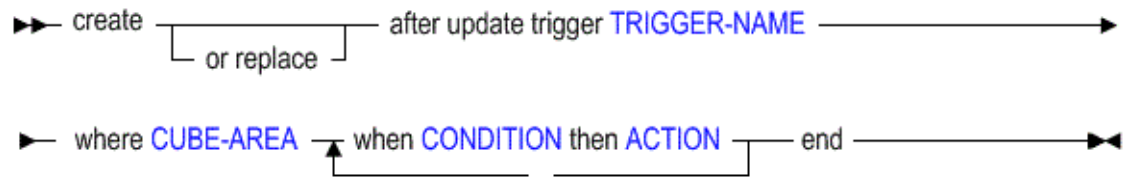
Note:

You cannot create or replace a trigger during a calculation, data update, or data load.

Note:

If a calculation assigns the same value to a given cell as was already present before the calculation, then triggers for that cell will not activate. In other words, if cell values are not changed, blocks are not marked as dirty, and triggers for those blocks are not activated, even if the trigger condition was otherwise met.

Syntax



- TRIGGER-NAME
- CUBE-AREA
- CONDITION
- ACTION

Keywords

Use the MaxL **create after update trigger** statement to create a trigger in the following ways:

create or replace after update trigger TRIGGER-NAME...

Create a new after-update trigger.

create or replace after update trigger TRIGGER-NAME...

Create an after-update trigger, or replace an existing trigger of the same name.

create after update trigger...where <cube area>...

The **where** clause is how you define the area of the cube to be tracked. Use a valid, symmetric MDX Slicer Specification.

create after update trigger...when <condition>...

The **when** clause is how you define the condition to be tested. Use the keyword **WHEN** followed by a valid MDX conditional expression. To test for a condition, you can use MDX conditional and logical operators, Boolean functions, and MDX iterative functions.

create after update trigger...then <action>...

The **then** clause is how you define the action to be taken if the **WHEN** condition is met. Refer to examples in [Examples of Triggers](#).

create after update trigger...end

The **END** keyword must terminate every create trigger statement.

Example

```
create or replace after update trigger Sample.Basic.WestColas where (Jan,
Sales, Actual, [100], West) when Jan < 1500 then spool WestColas_Fail end;
```

Create On-Update Trigger

The MaxL create on update trigger statement helps you create or replace a trigger to track state changes over a selected Essbase cube area. The trigger is activated immediately when a cell is updated.

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](#).

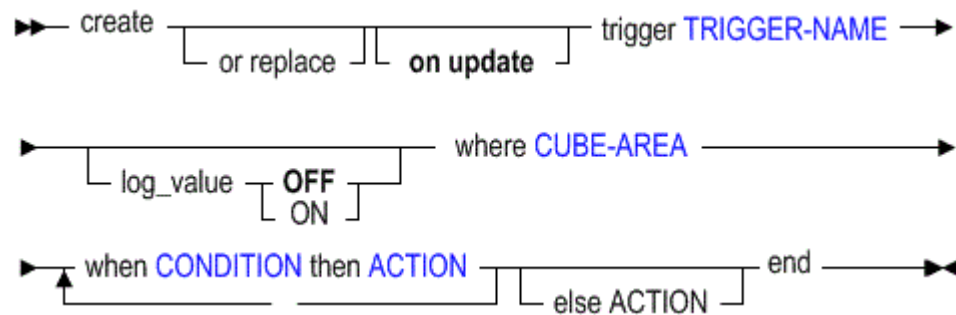
 **Note:**

If a calculation assigns the same value to a given cell as was already present before the calculation, then triggers for that cell will not activate. In other words, if cell values are not changed, blocks are not marked as dirty, and triggers for those blocks are not activated, even if the trigger condition was otherwise met.

 **Note:**

You cannot create or replace a trigger during a calculation, data update, or data load.

Syntax



- TRIGGER-NAME
- CUBE-AREA
- CONDITION
- ACTION

Keywords

Use the MaxL **create on update trigger** statement to create a trigger in the following ways:

create [on update] trigger TRIGGER-NAME...

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 TRIGGER-NAME...

Create an on-update trigger, or replace an existing trigger of the same name.

create [on update] trigger ... log_value OFF...

The optional **log_value OFF** syntax logs no data values to the trigger spool file. This is the default.

create [on update] trigger ... log_value ON...

The optional **log_value ON** syntax logs new and old data values to the trigger spool file.

create [on update] trigger ... where <cube area>...

The **where** clause is how you define the area of the cube to be tracked. Use a valid, symmetric MDX Slicer Specification.

create [on update] trigger ... when <condition>...

The **when** clause is how you define the condition to be tested. Use the keyword **WHEN** followed by a valid MDX conditional expression. To test for a condition, you can use MDX conditional and logical operators, Boolean functions, and MDX iterative functions.

create [on update] trigger ... then <action>...

The **then** clause is how you define the action to be taken if the **WHEN** condition is met. Refer to examples in [Examples of Triggers](#).

create [on update] trigger ... else <action>...

The optional **else** clause is a way to define an action to be taken if the **WHEN** condition is *not* met. Refer to examples in [Examples of Triggers](#).

create [on update] trigger ... end

The **END** keyword must terminate every **create trigger** statement.

Example

```
create or replace on update trigger Sample.Basic.EastColas log_value ON where
(Jan, Sales, Actual, [100], East) when Jan > 1850 then spool EastColas_alert
end;
```

Create User

Use MaxL to create an Essbase user.

This statement is supported for use in EPM Shared Services security mode only. You can only create users using **type external** grammar, to provision users that exist in EPM Shared Services.

Syntax

```
➤— create user USER-NAME type external —➤
```

USER-NAME

Use the MaxL **create user** statement to create a user in the following ways:

Keywords

create user USER-NAME type external

Create a user that must log in using the Enterprise Performance Management System security platform. For the user to log in successfully, the AUTHENTICATIONMODULE parameter must be set to CSS in the `essbase.cfg` file, and the name must match a valid user name in the external authentication repository.

Example

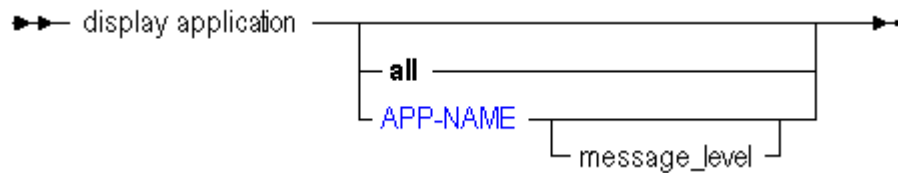
```
create user 'Autumn Smith' type external;
```

Creates a user called Autumn Smith who is externally authenticated in a corporate authentication repository supported by the EPM Shared Services security platform.

Display Application

The MaxL `display application` statement helps you view information about application-wide Essbase settings.

Syntax



APP-NAME

Use the MaxL **display application** statement to display application information in the following ways:

Keywords

display application [all]

Display all applications on the system.

Example:

```
display application;
```

display application APP-NAME

Display the named application.

Example:

```
display application Sample;
```

display application APP-NAME message_level

Display the message-level settings for the named application.

Example:

```
display application Sample message_level;
```

Sample output:

```
component          message_level
+-----+-----+
Sample            INFO
```

Output Columns

application

String. Name of the application.

comment

String. Optional description of the application.

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.

- [CALC-NAME](#)
- [DBS-NAME](#)
- [APP-NAME](#)

Use the MaxL **display calculation** statement to display calculations in the following ways:

Keywords

display calculation [all]

Display all stored calculations on the system.

Example:

```
display calculation;
```

display calculation CALC-NAME

Display the named calculation.

Example:

```
display calculation Sample.Basic.actuals;
```

display calculation on application

Display all calculations on the specified application.

Example:

```
display calculation on application Sample;
```

display calculation on database

Display all calculations on the specified database.

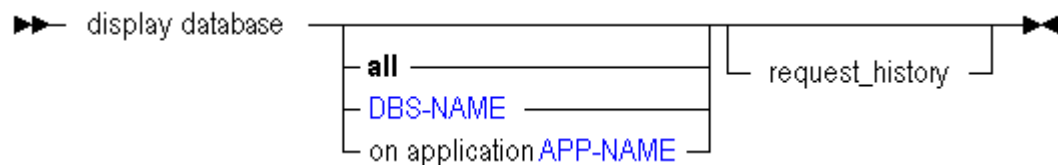
Example:

```
display calculation on database Sample.Basic;
```

Display Database

Use MaxL **display database** to view information about Essbase database-wide state, settings, and recent request history.

Syntax



- [DBS-NAME](#)
- [APP-NAME](#)

Keywords

Use the MaxL **display database** statement to display database information in the following ways:

display database [all]

Display information for all databases on the system.

Example:

```
display database;
```

display database DBS-NAME

Display information about the specified database (cube).

Example:

```
display database Sample.Basic;
```

display database on application

Display information about all databases on the specified application.

Example:

```
display database on application Sample;
```

display database ... request_history

Include in the database display information about recent requests for the database.

Information about the last three requests is returned.

Example:

```
display database on application ASOSamp request_history;
```

Output Columns Returned for MaxL Display Database

application

Name of the application.

database

Name of the database.

comment

Text of the database comment, if present.

startup

Whether the database is set to start when a user attempts retrievals against it.

autostartup

Whether the database is set to start when the application starts.

minimum permission

Minimum permission setting for the database.

Not applicable in EPM Shared Services security mode.

aggregate_missing

Whether Essbase aggregates missing values during database calculations.

two_pass_calc

Whether Two-Pass calculation is enabled.

create_blocks

Whether create blocks on equations is enabled.

data_cache_size

The size setting of the data cache for holding uncompressed data blocks.

file_cache_size

The size setting of the file cache.

index_cache_size

The size setting of the index cache, a buffer in memory that holds index pages.

index_page_size

The size setting for the index page, a subdivision of an index file that contains index entries that point to data blocks. This setting is not changeable.

cache_pinning

Whether cache memory locking is enabled (no longer supported).

compression

Compression type. Values returned for compression are numeric, and translate as follows:

1	Run-length encoding
2	Bitmap
3	(no longer supported)

retrieve_buffer_size

The size of the retrieval buffer, used to process and optimize retrievals from grid clients.

retrieve_sort_buffer_size

The size of the retrieval sort buffer, used to hold data to be sorted during retrievals.

io_access_mode

The current I/O access mode. Only buffered I/O is supported.

pending_io_access_mode

Values returned for pending_io_access_mode are numeric, and translate as follows:

0	Invalid / Error
1	Buffered
2	Direct /* no longer supported

no_wait

Whether Essbase is set to wait to acquire a lock on data blocks that are locked by another transaction.

committed_mode

Whether Essbase is set to enable transactions to hold read/write locks on all data blocks involved with a transaction until the transaction completes and commits.

pre_image_access

Whether Essbase is set to allow users read-only access to data blocks that are locked for the duration of another concurrent transaction.

lock_timeout

The maximum number of minutes that data blocks can be locked by users.

commit_blocks

The number of data blocks updated before Essbase performs a commit (The default is 3000).

commit_rows

The number of rows of a data file processed during a data load before Essbase performs a commit (The default is 0).

currency_database

Name of a linked currency database, if one exists.

currency_member

The member to use as a default value in currency conversions.

currency_conversion

The method of currency conversion.

Values returned for `currency_conversion` are numeric, and translate as follows:

1	division
2	multiplication

note

Annotation accessible from the login dialog box.

db_type

Database type. Values returned for `db_type` are numeric, and translate as follows:

0	Normal
1	Currency

read_only_mode

Values returned for `read_only_mode` are numeric, and translate as follows:

0	Not read only
1	Read only

db_status

Running status of the database. Values returned for `db_status` are numeric, and translate as follows:

0	Not Loaded
1	Loading
2	Loaded
3	Unloading

elapsed_time

How long the database has been running, in hours:minutes:seconds.

users_connected

Number of connected users.

blocks_locked

How many data blocks are locked.

number_dimensions

Number of dimensions.

number_disk_volume

Number of disk volumes.

data_status

Values returned for `data_status` are numeric, and translate as follows:

0	No Data
1	Data Loaded without Calculation
2	Data is Calculated

current_data_cache

Current size of the data cache.

current_file_cache

Current size of the file cache.

current_index_cache

Current size of the index cache.

current_index_page

Current size of the index page.

currency_country_dim

For currency databases, the country dimension.

currency_time_dim

For currency databases, the time dimension.

currency_category_dim

For currency databases, the accounts dimension where currency categories are defined.

currency_type_dim

For currency databases, the currency type dimension, which contains members that identify various currency scenarios.

request_type_n / request_user_n / request_start_n / request_end_n

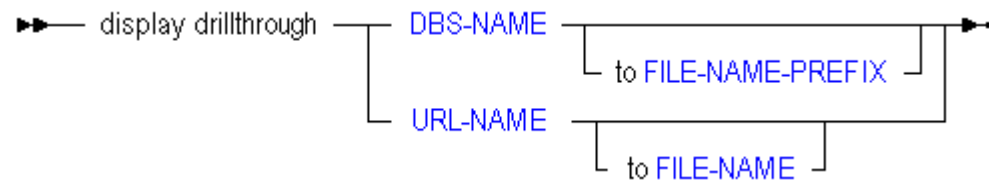
If you use the `request_history` keyword, information about the last three requests is returned under columns `request_type_n`, `request_user_n`, `request_start_n`, and `request_end_n`, where *n* is 1, 2, and 3. The `request_user` fields return the names of the users who made the requests. The `request_start` and `request_end` fields return the date and time of the requests. Values returned for `request_type` are numeric, and translate as follows:

0	Data Load
1	Calculation

Display Drillthrough

The MaxL `display drillthrough` statement helps you view drill-through URL definitions used to link Essbase to content hosted on Oracle ERP and EPM applications.

Syntax



- DBS-NAME
- FILE-NAME-PREFIX
- URL-NAME
- FILE-NAME

Use MaxL **display drillthrough** to display URL information in the following ways:

Keywords

display drillthrough DBS-NAME

Display all drill-through URL definitions on the database.
The number of drill-through URLs per database is limited to 255.
Example:

```
display drillthrough sample.basic;
```

display drillthrough 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.
Example:

```
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 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.

Example:

```
display drillthrough sample.basic."Drill through To EPMI";
```

Displays the drill-through URL definition named `Drill through To EPMI`.

display drillthrough 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."Drill through To EPMI" to  
"drillthrough.xml";
```

Displays the drill-through URL definition named `Drill through To EPMI`, writing the URL XML content to the file `drillthrough.xml`.

Display Filter

The MaxL display filter statement helps you view a specific Essbase security filter or a list of all filters.

Syntax

```

▶▶ display filter [all | FILTER-NAME]
                    on database DBS-NAME
  
```

- `FILTER-NAME`
- `DBS-NAME`

Use MaxL **display filter** to display filters in the following ways. Or, refer to [display filter row](#) if you need to display the access rows (contents) of a filter.

Keywords

display filter [all]

Display all filters on the system.

Example:

```
display filter;
```

display filter FILTER-NAME

Display a filter by name.

Example:

```
display filter Efficient.UserFilters.dslookupfilter;
```

display filter on database DBS-NAME

Display all filters associated with the specified database.

Example:

```
display filter on database Sample.Basic;
```

Display Filter Row

The MaxL display filter row statement helps you view the access definition rows in security filters that restrict Essbase database access.

Syntax

```
▶▶ display filter row [all | FILTER-NAME] on database DBS-NAME ◀◀
```

- **FILTER-NAME**
- **DBS-NAME**

You can display Essbase security filter contents in the following ways using MaxL **display filter row**.

Keywords

display filter row [all]

Display all filters (and their contents) defined on the system.

Example:

```
display filter row;
```

display filter row FILTER-NAME

Display a filter and its contents by name.

Example:

```
display filter row Sample.Basic.filt2;
```

display filter row on database DBS-NAME

Display all filters (and their contents) associated with the specified database.

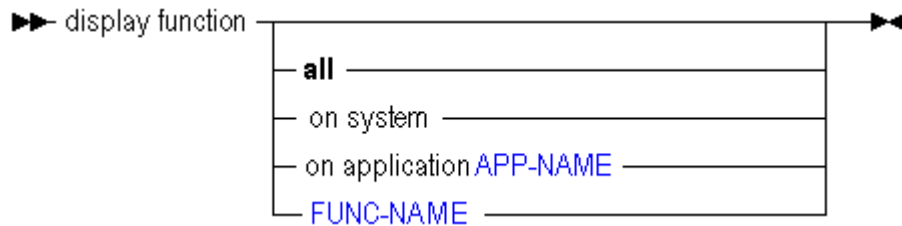
Example:

```
display filter row on Sample.Basic;
```

Display Function

The MaxL display function statement helps you view a list of custom-defined Essbase calculation functions (CDFs) available globally, or for an application. If MaxL shows no application name next to a function in the display output, then that macro is global. The minimum permission required to run this statement is Read.

Syntax



- APP-NAME
- FUNC-NAME

Keywords

Use the MaxL **display function** statement to display custom-defined functions (CDFs) in the following ways. The application must be loaded (started).

display function [all]

Display all custom-defined functions, including those registered on the application level (local) or on the system level (global).

Example:

```
display function;
```

display function on system

Display all custom-defined functions registered on the system (global). Does not include locally defined functions.

Example:

```
display function on system;
```

display function on application APP-NAME

Display all custom-defined functions registered with the specified application (local). Does not include globally defined functions.

Example:

```
display function on application Sample;
```

display function FUNC-NAME

Display a custom-defined function by name.

Example:

```
display function Sample.Basic.'@COVARIANCE';
```

Output Columns Returned for MaxL Display Function

The columns returned for **display function** are described as follows:

application

Application name(s).

function

Registered custom-defined function name(s), as defined by [FUNC-NAME](#) in the [create function](#) statement.

class

The java class before the method, as defined by [JAVACLASS.METHOD](#) in the create function statement.

method

The java method (at the end of the class), as defined by [JAVACLASS.METHOD](#) in the create function statement.

spec

Optional Essbase calculator-syntax specification string, as defined by [CALC-SPEC-STRING](#) in the create function statement.

comment

String as defined by [COMMENT-STRING](#) 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.

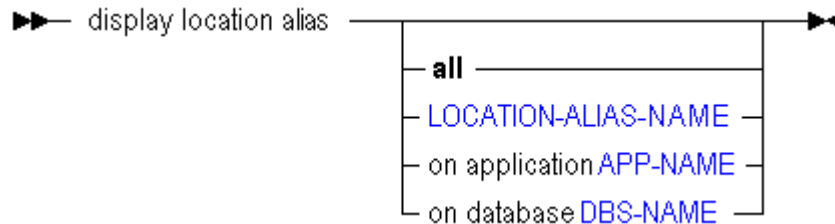
Values:

- 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.

Display Location Alias

The MaxL display location alias statement helps you view one or more defined Essbase location aliases used to reference other cubes. You can view a specific location alias or a list of all location aliases on the Essbase Server.

Syntax



- LOCATION-ALIAS-NAME
- APP-NAME
- DBS-NAME

Keywords

You can display location aliases in the following ways using MaxL **display location alias**.

display location alias [all]

Display all location aliases defined on the Essbase Server. To display all the location aliases, all the cubes on the Essbase Server must be started.

Example:

```
display location alias;
```

display location alias LOCATION-ALIAS-NAME

Display a location alias by name. To use this statement, the cube must be started first.

Example:

```
display location alias Sample.Basic.EasternDB;
```

display location alias on application APP-NAME

Display all location aliases defined for the specified application. To display all the location aliases on the application, all cubes on the application must be started first.

Example:

```
display location alias on application Sample;
```

display location alias on database DBS-NAME

Display all location aliases defined for the specified cube. To use this statement, the cube must be started first.

Example:

```
alter application Sample load database Basic;
display location alias on database Sample.Basic;
```

Display Lock

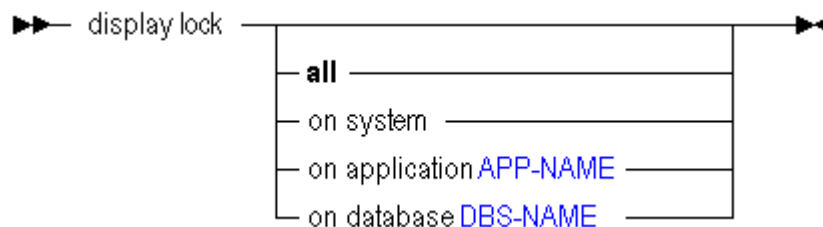
The MaxL display lock statement helps you view information about current locks held by users or processes against Essbase data blocks in a block storage (BSO) cube.



Note:

Data locks do not apply to aggregate storage cubes.

Syntax



- [APP-NAME](#)
- [DBS-NAME](#)

You can display Essbase data locks in the following ways using MaxL **display lock**.

Keywords

display lock [all]

Display all locks on the specified scope. If **all** is omitted, this is the default.

Example:

```
display lock;
```

display lock on system

Display all locks on the Essbase Server.

Example:

```
display lock on system;
```

display lock on application APP-NAME

Display all locks associated with the specified application.

Example:

```
display lock on application Sample;
```

display lock on database DBS-NAME

Display all locks associated with the specified database.

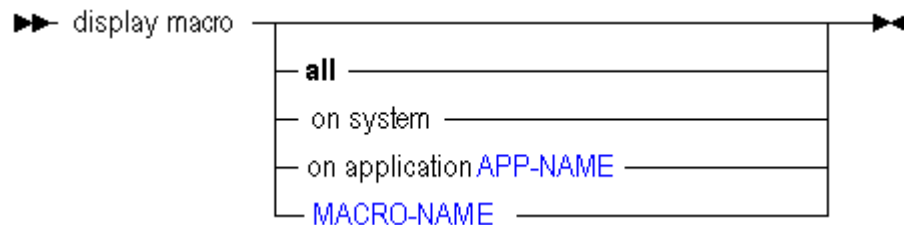
Example:

```
display lock on database Sample.Basic;
```

Display Macro

The MaxL display macro statement helps you view a list of Essbase custom-defined calculation macros (CDMs) available globally, or for an application. If MaxL shows no application name next to a macro in the display output, then that macro is global. The minimum permission required to run this statement is Read.

Syntax



- APP-NAME
- MACRO-NAME

Keywords

You can display custom-defined macros in the following ways using MaxL **display macro**. The application must be loaded (started).

display macro [all]

Display all custom-defined macros, including those registered on the application level (local) or on the system level (global).

Example:

```
display macro;
```

display macro on system

Display all custom-defined macros registered on the Essbase Server (globally). Does not include locally defined macros.

Example:

```
display macro on system;
```


display macro on application APP-NAME

Display all custom-defined macros registered with the specified application (locally). Does not include globally defined macros.

Example:

```
display macro on application Sample;
```

display macro MACRO-NAME

Display a custom-defined macro by name.

Example:

```
display macro Sample.'@COUNTRANGE';
```

Output Columns Returned for MaxL Display Macro

The columns returned for this statement are described as follows:

application

Application name(s).

macro

Macro name(s), as defined by [MACRO-NAME](#) 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](#) in the [create macro](#) statement.

spec

Optional Essbase calculator-syntax specification string, as defined by [CALC-SPEC-STRING](#) in the [create macro](#) statement.

comment

String as defined by [COMMENT-STRING](#) in the [create macro](#) statement.

state

The current state of the registered custom-defined macro.

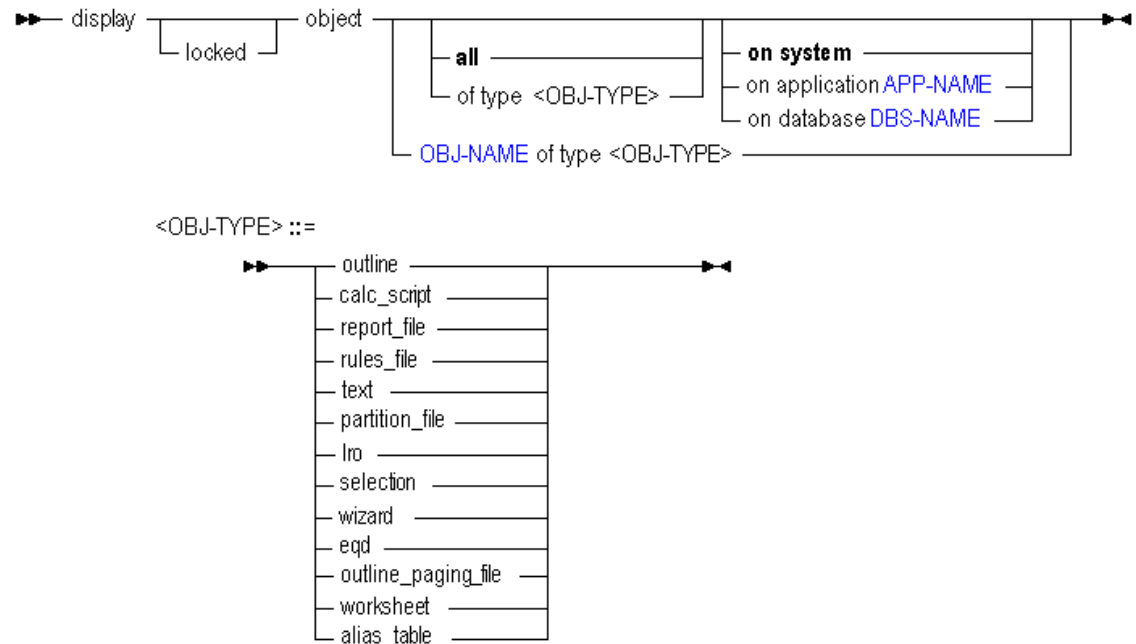
Values:

- 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.

Display Object

The MaxL display object statement helps you view one or more Essbase file objects stored in cube directories.

Syntax



- APP-NAME
- DBS-NAME
- OBJ-NAME

Keywords

You can display objects in the following ways using MaxL **display object**.

display object [all]

Display all stored objects on the specified scope.

Example:

```
display object;
```

display locked object...

Display only locked objects on the specified scope.

Example:

```
display locked object on application Sample;
```

display object of type OBJ-TYPE

Display only the objects of a specific type. Valid object types (OBJ-TYPE) in Essbase 21c are: outline, calc_script, report_file, rules_file, text, partition_file, lro, and worksheet.

To display alias tables, use query `database appname.dbname list alias_table` instead of **display object**.

Example:

```
display object of type calc_script;
```

display object OBJ-NAME of type OBJ-TYPE

Display a specific object by name and type.

Example:

```
display object sample.basic.Calcdat of type text;
```

display object ...on system

Display all objects, or all specified objects, that exist on the Essbase Server.

Example:

```
display object of type rules_file on system;
```

display object ...on application APP-NAME

Display objects associated with the specified application. Does not include database objects.

Example:

```
display object all on application Sample;
```

display object ...on database DBS-NAME

Display objects associated with the specified database.

Example:

```
display object of type text on database Sample.Basic;
```

Example Including Output

```
MAXL> display object sample.basic.Data_Basic of type text;
```

application	database	object_name	object_type	locked
locked_by	locked_time			
Sample	Basic	Data_Basic	9	FALSE
N/A	N/A			

The **object_type** output column is numeric, and the values translate as follows:

```
1  outline
2  calc_script
3  report_file
4  rules_file
8  worksheet
9  text
10 partition_file
11 lro
```

Display Partition

The MaxL display partition statement helps you view information about Essbase partitioned databases. This statement only displays partition information for applications which are currently started.

Syntax



DBS-NAME

You can display partition information in the following ways using MaxL **display partition**.

Keywords

display partition [all]

Display all partitions defined on the Essbase Server.

Example:

```
display partition;
```

display partition on database DBS-NAME

Display all partitions associated with the specified database.

Example:

```
display partition on database Sample.Basic;
```

display partition ... advanced

Display full information including areas and member mappings for local and remote pieces of partitions.

Example:

```
display partition on database Sample.Basic advanced;
```

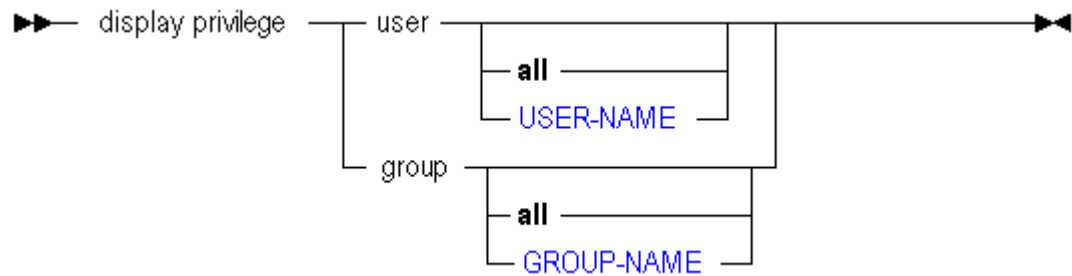
Notes

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.

Display Privilege

The MaxL display privilege statement helps you view Essbase assigned privileges, calculations, and filters held by users and groups.

Syntax



- USER-NAME
- GROUP-NAME

You can display Essbase security permissions in the following ways using MaxL **display privilege**.

Keywords

display privilege user [all]

Display security permissions for all users.

Example:

```
display privilege user Fiona;
```

display privilege user USER-NAME

Display security permissions for all users, or for a specified user.

display privilege group [all]

Display security permissions for all groups.

Example:

```
display privilege group;
```

display privilege group GROUP-NAME

Display security permissions for a specified group.

Example:

```
display privilege group CalcGrp;
```

MaxL Display Privilege Output Columns

The following is an example of the output columns returned by the **display privilege** statement.

holder	application	database	resource	type
sample_basic	Sample	Basic	actuals	3
NewGrp	Sample		no_access	2
FilterGroup	Sample	Basic	filter1	4
FilterGroup	Sample	Basic	read	2
powergroup			create_appli	1
...				

The numeric values returned for the *type* field can be interpreted as follows:

Table 3-2 Output Columns for Display Privilege: Type Field

Type Value	Type Description
1	System-Level System Privileges
2	System-Level System Roles
3	Execute calculation
4	Filter

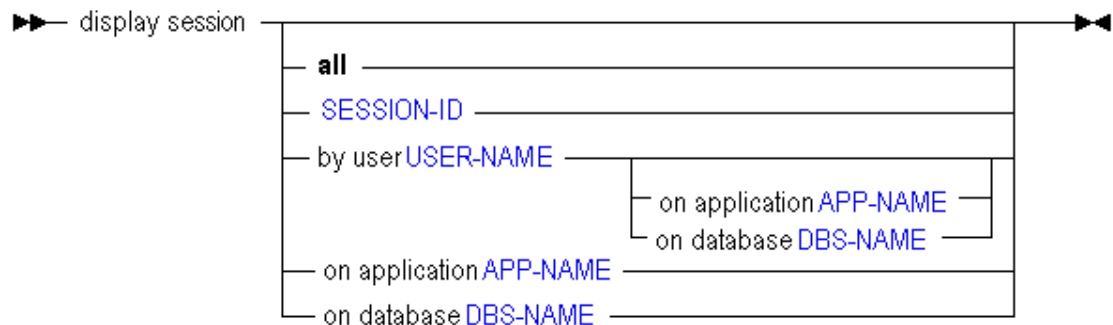
Display Session

The MaxL display session statement helps you view active login sessions on the current Essbase Server, application, or database.

The information that **display session** returns about active login sessions includes:

- 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



- APP-NAME
- DBS-NAME
- USER-NAME
- SESSION-ID

Keywords

You can display login and request information in the following ways using MaxL **display session**.

display session [all]

Display information about all current user sessions and active requests.

Example:

```
display session;
```

display session SESSION-ID

Display information about a particular user session, indicated by the numeric session ID.

Example:

```
display session 865075202;
```

display session by user USER-NAME

Display information about all current sessions by a particular user.

Example:

```
display session by user jbrownst;
```

display session by user USER-NAME on application APP-NAME

Display information about all current sessions by a particular user on the specified application.

Example:

```
display session by user jbrownst on application Sample;
```

display session by user USER-NAME on database DBS-NAME

Display information about all current sessions by a particular user on the specified database.

Example:

```
display session by user jbrownst on database Sample.Basic;
```

display session on application APP-NAME

Display information about all current sessions on the specified application.

Example:

```
display session on application Sample;
```

display session on database DBS-NAME

Display information about all current sessions on the specified database.

Example:

```
display session on database Sample.Basic;
```

Output Columns Returned for MaxL Display Session

user

Logged in user name. Example: powerusr.

session

Numeric session id. Example: 865075202.

login_time

Number of seconds ago the session began. Example: 192.

application

Name of active application. Example: `Sample`.

database

Name of active cube. Example: `Basic`.

db_connect_time

Number of seconds ago the cube was set active. Example: `11879`.

request

Type of active request in progress; for example, calculation, data load, or restructure. This information can help you get details about what is occurring during lengthy sessions. Example: `BuildDimXml : Index Only`.

request_time

Number of milliseconds the active request has been running. Example: `1503869494621`.

connection_source

Host name of the connected service. Example: `example.com`.

connection_ip

IP address of the connected service. Example: `192.0.2.123`.

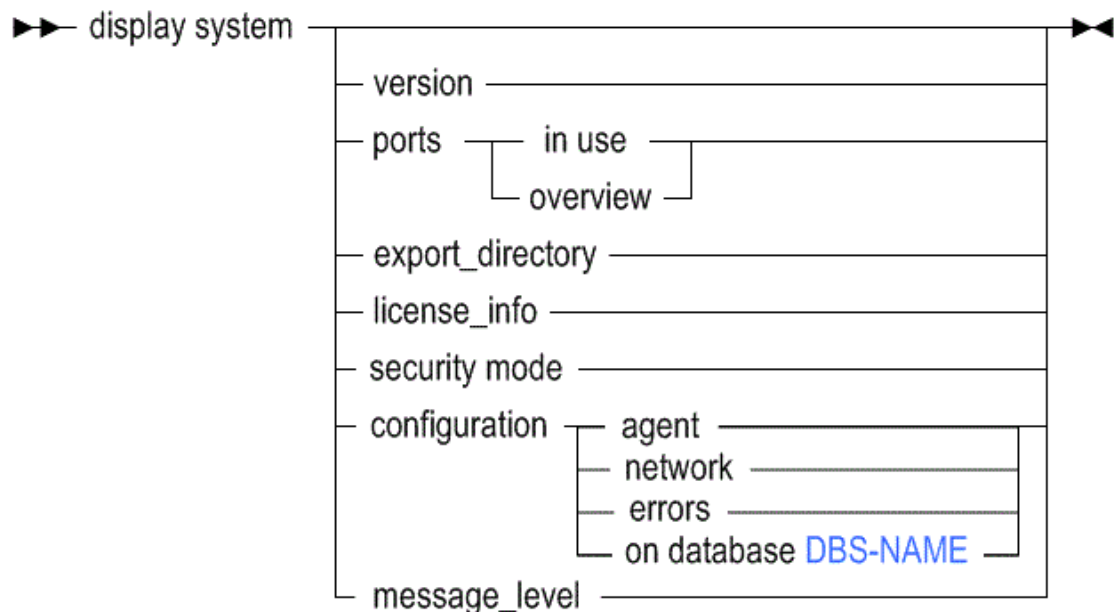
request_state

The status of the active request. Example: `in_progress`.

Display System

The MaxL display system statement helps you view information about current server-wide Essbase settings.

Syntax



`DBS-NAME`

Keywords

You can display Essbase Server information in the following ways using MaxL **display system**.

display system

Display current connections and Essbase Server-wide settings.

Example:

```
display system;
```

In the display output, the **configuration** field values are numeric, and translate as follows:

```
1 Non-Unicode mode
2 Unicode mode
```

display system version

Display the Essbase Server software version number.

Example:

```
display system version;
```

display system ports in use

Display information about ports currently in use on the Essbase Server.

Example:

```
display system ports in use;
```

display system ports overview

Display the number of ports that are available and in use on the Essbase Server.

Example:

```
display system ports overview;
```

display system export_directory

Display names of directories created for linked-reporting objects exported from a database to a directory created in the application directory.

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 application directory using the following **export lro** method are listed:

```
export database DBS-NAME lro to <server or local> directory DBS-EXPORT-DIR;
```

where DBS-EXPORT-DIR is a suffix (for example, `dir1`) for the name of a directory created by MaxL in the application directory. MaxL creates the directory with a prefix of `appname-dbsname-`. For example, **display system export_directory** would list the following directories existing under the application directory:

```
sample-basic-dir1
```

```
sample-basic-dir2
```

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`

Example:

```
display system export_directory;
```

display system license_info

Display information about the license settings implemented on the Essbase Server.

Example:

```
display system license_info;
```

display system security mode

The type of security in use: native or OPSS mode.

Example:

```
display system security mode;
```

In the output for this statement, **security_mode** field values are numeric, and translate as follows:

- 1 Native Essbase security (no longer supported)
- 2 OPSS security

display system configuration agent

Display current Essbase Agent configuration properties.

Permission required: Administrator.

Example:

```
display system configuration agent;
```

display system configuration network

Display current Essbase Server configuration properties applicable to the network.

Permission required: Administrator.

Example:

```
display system configuration network;
```

display system configuration errors

Display Essbase configuration properties that contain errors: an error is any line entry that is not a comment *and* results in nothing being set.

Permission required: Administrator.

Example:

```
display system configuration errors;
```

display system configuration on database DBS-NAME

Display Essbase configuration properties applicable to the named database.

Permission required: Administrator.

Example:

```
display system configuration on database Sample.Basic;
```

display system message_level

Display the values that are set for the Essbase Server message level.

Example:

```
display system message_level;
```

The output lists the following information:

```
component          message_level
+-----+-----+
system             info
```

Sample Outputs for MaxL Display System Configuration

```
MAXL> set column_width 40;
```

```
MAXL> display system configuration agent;
```

```
KEYWORDS          SETTINGS
+-----+-----+
AGENTTHREADS      50
MAXLOGINS          100000
PORTUSAGELOGINTERVAL 600
```

OK/INFO - 1241044 - Records returned: [3].

```
MAXL> display system configuration network;
```

```
KEYWORDS          SETTINGS
+-----+-----+
NETDELAY          1500
NETRETRYCOUNT    2000
```

OK/INFO - 1241044 - Records returned: [2].

```
MAXL> display system configuration on database democfg.basic;
```

```
KEYWORDS          SETTINGS
+-----+-----+
CALCCACHE         TRUE
CALCCACHEDEFAULT 1250000
CALCCACHEHIGH     1750000
CALCCACHELOW      40000
DLSINGLETHREADPERSTAGE FALSE
DLTHREADSPREPARE  4
DLTHREADSWRITE    4
DYNCALCCACHEMAXSIZE DB[41943040], SV[41943040]
SSPROCROWLIMIT    250000
```

OK/INFO - 1241044 - Records returned: [9].

Display Trigger

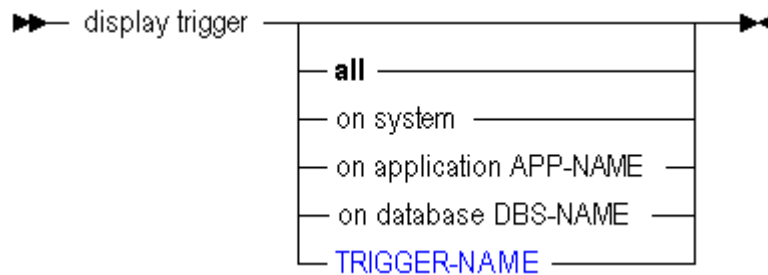
The MaxL display trigger statement helps you view details about triggers that track changes to selected Essbase cube areas.



Note:

The application containing the trigger must be started in order to use this statement.

Syntax



APP-NAME

Keywords

You can display triggers in the following ways using MaxL **display trigger**.

display trigger [all]

Display all triggers defined on the Essbase Server.

Example:

```
display trigger;
```

display trigger on system

Same as **display trigger all**.

Example:

```
display trigger on system;
```

display trigger on application APP-NAME

Display all triggers defined for the specified application.

Example:

```
display trigger on application Sample;
```

display trigger on database DBS-NAME

Display all triggers defined for the specified cube.

Example:

```
display trigger on database Sample.Basic;
```

display trigger TRIGGER-NAME

Display a trigger by name.

Example:

```
display trigger Sample.Basic.WestColas;
```

Table 3-3 Output Columns Returned for MaxL Display Trigger

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 .

Display Trigger Spool

The MaxL display trigger spool statement helps you view log files generated by triggers that track changes to selected Essbase cube areas.

For more information about triggers, see [Examples of Triggers](#).

Syntax

```

▶▶ display trigger_spool [all | on application APP-NAME | on database DBS-NAME]
SPOOL-NAME ◀◀

```

- APP-NAME
- DBS-NAME
- SPOOL-NAME

Keywords

You can display trigger output (spool) files in the following ways using MaxL **display trigger_spool**.

display trigger_spool [all]

Display all triggers defined on the Essbase Server.

Example:

```
display trigger_spool;
```

display trigger_spool on application APP-NAME

Display all trigger spools for the specified application.

Example:

```
display trigger_spool on application Sample;
```

display trigger_spool on database DBS-NAME

Display all trigger spools for the specified cube.

Example:

```
display trigger_spool on database Sample.Basic;
```

display trigger_spool SPOOL-NAME

Display a trigger spool by name.

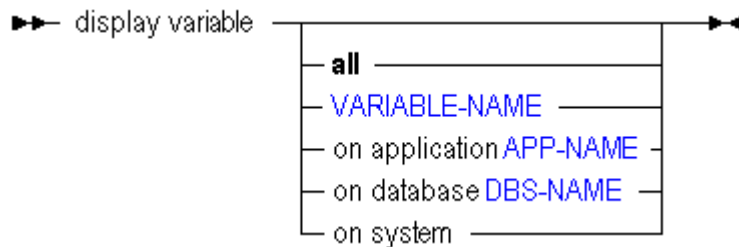
Example:

```
display trigger_spool Sample.Basic.EastColas_alert;
```

Display Variable

The MaxL display variable statement helps you view a list of Essbase substitution variables defined on the server.

Syntax



- VARIABLE-NAME
- APP-NAME
- DBS-NAME

Keywords

You can display substitution variables in the following ways using MaxL **display variable**.

display variable [all]

Display all substitution variables defined on the Essbase Server, including those associated with applications and databases.

Example:

```
display variable;
```

display variable VARIABLE-NAME

Display a substitution variable by name.

The minimum permission required to display variables is:

- Read access, to display variables associated with the specified database or application.
- Administrator, to display system-defined variables.

Example:

```
display variable Sample.curmonth;
```

display variable on application APP-NAME

Display only substitution variables defined on the specified application.

The minimum permission required to run this statement is Read access for the specified application.

Example:

```
display variable on application Sample;
```

display variable on database DBS-NAME

Display only substitution variables defined on the specified database.

The minimum permission required to run this statement is Read access for the specified database.

Example:

```
display variable on database CalcTuple.Tuple;
```

display variable on system

Display only the substitution variables associated with the Essbase Server.

The permission required to run this statement is Administrator.

Example:

```
display variable on system;
```

Notes

To create and manage substitution variables, use MaxL [alter database](#) (containing add, drop, and set variable).

Drop Application

The MaxL drop application statement helps you delete an application from the Essbase Server. To remove an application with databases, use **cascade**. To remove an application that has locked objects in a constituent database, you can use **force**. The minimum permission required to remove an application is Application Manager.

Syntax

```

▶▶ drop application APP-NAME [ cascade ] [ force ] ▶▶

```

APP-NAME

You can delete applications in the following ways using **drop application**.

Keywords**drop application APP-NAME**

Delete an empty application by name.

Example:

```
drop application Sample;
```

drop application APP-NAME cascade

Delete an application along with its constituent databases.

Example:

```
drop application Sample cascade;
```

drop application APP-NAME force

Delete an application that may have locked objects in a constituent database.

Example:

```
drop application Sample cascade force;
```

Drop Calculation

The MaxL drop calculation statement helps you delete a stored calculation (calc script) from an Essbase database. The minimum permission required to remove a calc script is Database Manager.

Syntax

```

▶▶ drop calculation CALC-NAME ▶▶

```

CALC-NAME**Keywords****drop calculation CALC-NAME**

Delete the specified calculation.

Example

```
drop calculation Sample.basic.calcname;
```


Removes a calculation from Sample.basic.

Drop Database

The MaxL drop database statement helps you delete an Essbase database. If the database has outstanding locks, you can clear them first, or use the **force** keyword to drop with locks. The minimum permission required to remove a database is Database Manager.

Syntax

```

▶▶ drop database DBS-NAME ───────────▶▶
      └── force ───┘
  
```

DBS-NAME

Keywords

drop database DBS-NAME

Delete a database (cube) by name.

drop database DBS-NAME force

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

The MaxL drop drillthrough statement helps you delete a drill-through URL definition used to link from the Essbase cube to Web content hosted on Oracle ERP and EPM applications.

Syntax

```

▶▶ drop drillthrough URL-NAME ─────────▶▶
  
```

URL-NAME

Example

```
drop drillthrough sample.basic.myURL;
```

Drop Filter

The MaxL drop filter statement helps you delete a security filter from an Essbase cube. The minimum permission required to remove a filter is Database Manager.

Syntax

```
▶▶ drop filter FILTER-NAME ◀◀
```

FILTER-NAME

Keywords

drop filter FILTER-NAME

Delete a filter by name.

Example

```
drop filter sample.basic.filter1;
```

Removes the filter called filter1 from the Sample Basic database.

Drop Function

The MaxL drop function statement helps you delete (de-register) a custom-defined Essbase calculation function (CDF) from the Essbase Server or from an application.

The minimum permission required to run this statement is:

- Application Manager, to remove a local (application-level) function.
- System Administrator, to remove a global (Essbase Server-level) function.

Syntax

```
▶▶ drop function FUNC-NAME ◀◀
```

FUNC-NAME

Keywords

You can remove (de-register) custom-defined calculation functions from Essbase using **drop function**.

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';
```

Removes the function called @COVARIANCE from the Sample application.

Drop Group

When user authentication is managed either through EPM Shared Services or an external LDAP identity provider, you can use the MaxL drop group statement if you need to clean up inactive groups from Essbase after they have been removed or renamed on the external provider.

This MaxL statement is no longer supported for any other use than the one stated above.

Syntax

```
▶▶ drop group GROUP-NAME ◀◀
```

GROUP-NAME

You can clean up security groups from Essbase using drop group.

Keywords

drop group <group-name>

Delete a group from Essbase. This action does not de-provision the group from the external identity provider.

Example

```
drop group test_group;
```

Removes the group called test_group from Essbase.

Drop Location Alias

The MaxL drop location alias statement helps you delete a location alias defined on an Essbase database. Location aliases are used to reference another Essbase database. The minimum permission required to run this statement is Database Manager.

Syntax

▶▶ — drop location alias **LOCATION-ALIAS-NAME** —▶▶

LOCATION-ALIAS-NAME

Keywords

drop location alias <location-alias-name>

Delete a location-alias definition by name.

Example

```
drop location alias Main.Sales.EasternDB;
```

Removes the location alias called EasternDB in the Main.Sales database.

Drop Lock

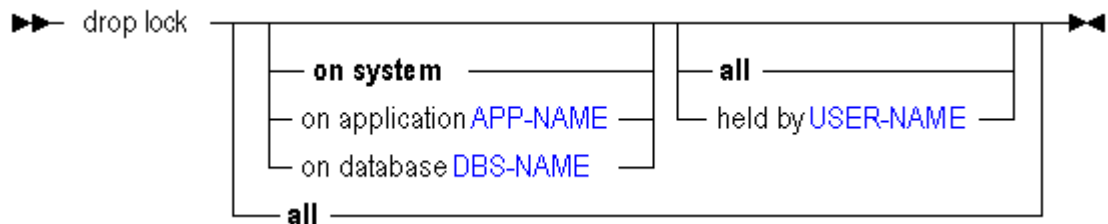
The MaxL drop lock statement helps you remove a lock acquired through a Smart View or other grid client operation against an Essbase block storage (BSO) database.



Note:

Data locks do not apply to aggregate storage (ASO) applications.

Syntax



- **APP-NAME**
- **USER-NAME**
- **DBS-NAME**

Keywords

You can delete locks in the following ways using MaxL **drop lock**.

drop lock [on system] [all]

Drops all locks by all users, for all block storage databases on the Essbase Server.

Example:

```
drop lock;
```

drop lock all

Same as "drop lock on system all," removes all locks, by all users, for all block storage cubes on the Essbase Server.

Example:

```
drop lock all;
```

drop lock on system

Same as "drop lock on system all," deletes all locks for all block storage cubes on the Essbase Server.

Example:

```
drop lock on system;
```

drop lock on application APP-NAME

Drops all locks from block storage cubes on the specified application, for all users.

Example:

```
drop lock on application Sample;
```

drop lock on application APP-NAME held by USER-NAME

Drops any locks, on block storage cubes in the named application, which are held by a specific user.

Example:

```
drop lock on application Sample held by user adparvan;
```

drop lock on database DBS-NAME

Drops all locks on the specified block storage database, held by any and all users.

Example:

```
drop lock on database Sample.Basic all;
```

drop lock on database DBS-NAME held by USER-NAME

Drops locks on the specified block storage database that are held by a specific user.

Example:

```
drop lock on database Sample.Basic held by krsandrs;
```

drop lock held by USER-NAME

Drops all locks held by a specific user.

Example:

```
drop lock held by lydsimmons;
```

Drop Macro

The MaxL drop macro statement helps you remove a custom-defined Essbase calculation macro (CDM).

The minimum permission required to delete a custom-defined macro is:

- Application Manager, to drop a local (application-level) macro.
- System Administrator, to drop a global (system-level) macro.

Syntax

```
▶▶ drop macro MACRO-NAME ◀◀
```

MACRO-NAME

Keywords

drop macro MACRO-NAME

Delete a custom-defined macro by name.

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

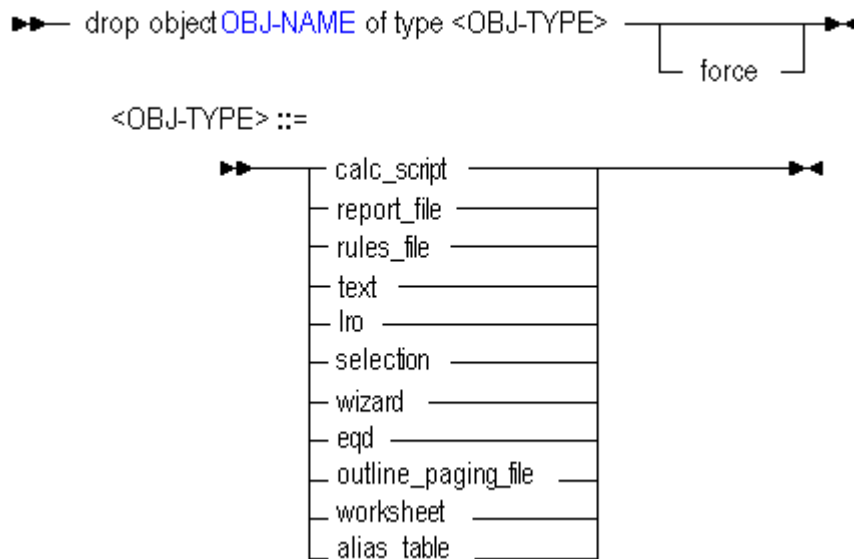
```
drop macro sample.'@COVARIANCE';
```

Deletes the macro called @COVARIANCE from the Sample application.

Drop Object

The MaxL drop object statement helps you remove Essbase database-related objects from the file catalog.

Syntax



OBJ-NAME

Keywords

You can delete objects in the following ways using MaxL **drop object**.

drop object OBJ-NAME of type OBJ-TYPE

Delete a specified Essbase cube artifact. You must specify the object name and type.

drop object ...force

If the object is locked by a user or process, you can use the **force** keyword if necessary, to unlock and delete it.

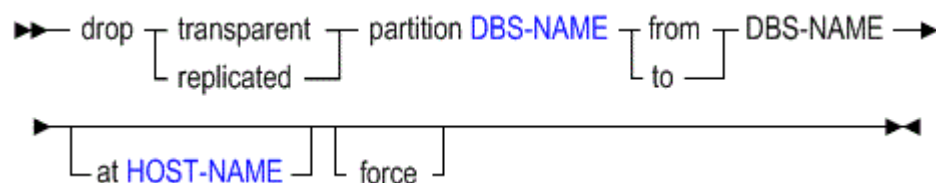
Notes

To drop a partition, use [drop partition](#) instead of **drop object**.

Drop Partition

The MaxL drop partition statement helps you remove a partition definition between two Essbase databases. To use this statement, Database Manager permission for each cube is required.

Syntax



- **DBS-NAME**
- **HOST-NAME**

Keywords

You can delete partition definitions in the following ways using MaxL **drop partition**.

drop...partition DBS-NAME from...

Remove a partition definition between the current target cube and a source cube.

drop...partition DBS-NAME to...

Remove a partition definition between the current source cube and a target cube.

drop...partition ... at <host-name>

Specify the host location if you are removing a partition definition associated with a remote instance.

Use the discovery URL (ending in **/agent**) to indicate the location. For example: "https://myserver.example.com:9001/essbase/agent"

drop...partition ... force

Use the **force** keyword if you need to drop the source half of a partition definition even if the target half is missing or invalid. For more information, see [Force Deletion of Partitions](#).

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(Qtr1)' to samppart.company at
 "https://myserver.example.com:9001/essbase/agent";

drop replicated partition Samppart.Company from Sampeast.East;
```

Drop Trigger

The MaxL drop trigger statement helps you remove a trigger created to track state changes over a selected Essbase cube area.

Syntax

```
▶▶ drop trigger TRIGGER-NAME ─────────▶▶▶
```

TRIGGER-NAME

Example

```
drop trigger Sample.Basic.WatchCosts ;
```

Drop Trigger Spool

The MaxL `drop trigger_spool` statement helps you delete the log file created by an Essbase trigger.

Triggers track state changes over a selected cube area. For more information about triggers, see [Examples of Triggers](#).

Syntax

```
drop trigger_spool SPOOL-NAME  
                  all on database DBS-NAME
```

- SPOOL-NAME
- DBS-NAME

Keywords

You can delete trigger log files following ways using MaxL **drop trigger_spool**.

drop trigger_spool SPOOL-NAME

Delete the spool file by name.

Example:

```
drop trigger_spool Sample.Basic.spoolfile;
```

drop trigger_spool all on database DBS-NAME

Delete all the spool files on a specific cube.

Example:

```
drop trigger_spool all on database Sample.Basic;
```

Drop User

When user authentication is managed either through EPM Shared Services or an external LDAP identity provider, you can use the MaxL `drop user` statement if you need to clean up inactive users from Essbase after they have been removed or renamed on the external provider.

This MaxL statement is no longer supported for any other use than the one stated above.

Syntax

```
drop user USER-NAME
```

USER-NAME

Keywords

You can clean up users from Essbase using MaxL `drop user`.

drop user USER-NAME

Delete a user from Essbase. This action does not de-provision the user from the external identity provider.

Example

```
drop user wcrane;
```

Execute Calculation

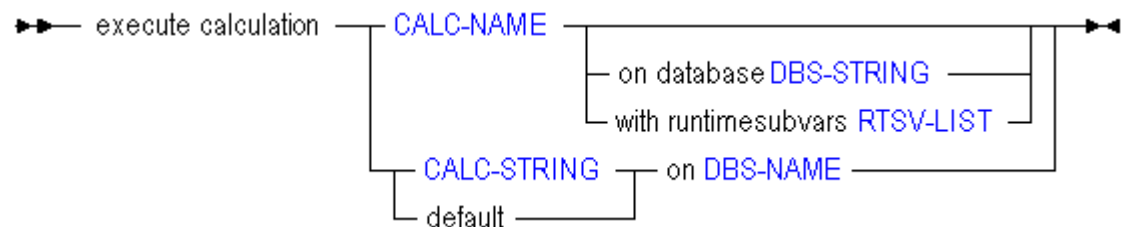
The MaxL **execute calculation** statement helps you calculate an Essbase block storage database. You can use this statement to run a stored calculation script, a calculation string, or the default calc.

[Click here for aggregate storage version](#)

Using this statement, you can execute a particular stored calculation, or the default stored calculation (determined by [alter database](#)), or an anonymous (non-stored) calculation string.

The minimum permission required to run a stored BSO calculation is having that calc script provisioned to you. To run an anonymous calculation (CALC-STRING method) or the default calculation, the minimum application permission required is Database Update.

Syntax



- CALC-NAME
- DBS-STRING
- RTSV-LIST
- CALC-STRING
- DBS-NAME

Keywords

You can run BSO calc scripts in the following ways using MaxL **execute calculation**.

execute calculation CALC-NAME

Run the specified stored calculation script.

Example:

```
execute calculation Sample.Basic.Calc1;
```

Calculates the Sample.Basic database using the stored calculation script file named Calc1, which is associated with the database.

execute calculation CALC-NAME on database DBS-STRING

Run the specified stored calculation script against the specified database.

Example:

```
execute calculation Sample.Calc2 on database Basic;
```

Calculates the Sample.Basic database using the stored calculation script file named Calc2, which is associated with the Sample application.

execute calculation CALC-STRING on DBS-NAME

Run an anonymous calculation, whose body is contained in *CALC-STRING*, against the specified database.

Example:

```
execute calculation 'SET MSG ERROR; CALC ALL;' on Sample.Basic;
```

Calculates the Sample.Basic database using an anonymous (unstored) calculation string.

execute calculation default on DBS-NAME

Run the default calculation against the specified database.

Example:

```
execute calculation default on Sample.Basic;
```

execute calculation CALC-NAME with rtimesubvars RTSV-LIST

Run the specified stored calculation script with the runtime substitution variables specified in *RTSV-LIST*, which is a string of runtime substitution variables specified as key/value pairs. The string must be enclosed with single quotation marks, and the key/value pairs must be separated by a semicolon, including a semicolon after the last runtime substitution variable in the string and before the terminal single quotation mark. In this example of a runtime substitution variable string, the name and value of four runtime substitution variables are specified (for example, the value of the runtime substitution variable named "a" is 100):

```
'a=100;b=@CHILDREN("100");c="Actual"->"Final";d="New York";'
```

The string of runtime substitution variables cannot exceed 64 KB.

Example:

```
execute calculation Sample.Basic.Calc3 with rtimesubvars 'a=100;b=50;';
```

Calculates the Sample.Basic database using the stored calculation script file named Calc3, which is associated with the database, and the specified runtime substitution variables, in which the value of the runtime substitution variable named "a" is 100 and the value of "b" is 50.

 **Note:**

Runtime substitution variables used in a calculation script must be declared in the SET RUNTIMESUBVARS calculation command, with a name and default value. If a different value is declared in the *RTSV-LIST*, the default value is overwritten at runtime.

If you include a runtime substitution variable in *RTSV-LIST* that has not been declared in SET RUNTIMESUBVARS, Essbase ignores the undeclared runtime substitution variable (no warnings or exceptions are generated).

Runtime substitution variables that are used in a calculation script can be logged in the application log file, using the ENBLERTSVLOGGING configuration setting.

If the name of a runtime substitution variable that is declared in the SET RUNTIMESUBVARS calculation command is the same as a runtime substitution variable declared in *RTSV-LIST*, the value specified in *RTSV-LIST* overwrites the default value in SET RUNTIMESUBVARS.

Notes

- A stored calculation can be associated with a specific database in an application (database level), or with an application only (application level). To execute a calculation stored at the application level, you must specify which database in the application to calculate using the **on database STRING** grammar.
- A calculation script can reference runtime substitution variables using the **with runtimesubvars** grammar.

See Also

Runtime Substitution Variables in Calculation Scripts Run in Essbase

SET RUNTIMESUBVARS

ENBLERTSVLOGGING

Export Data

The MaxL **export data** statement helps you export data from an Essbase database.

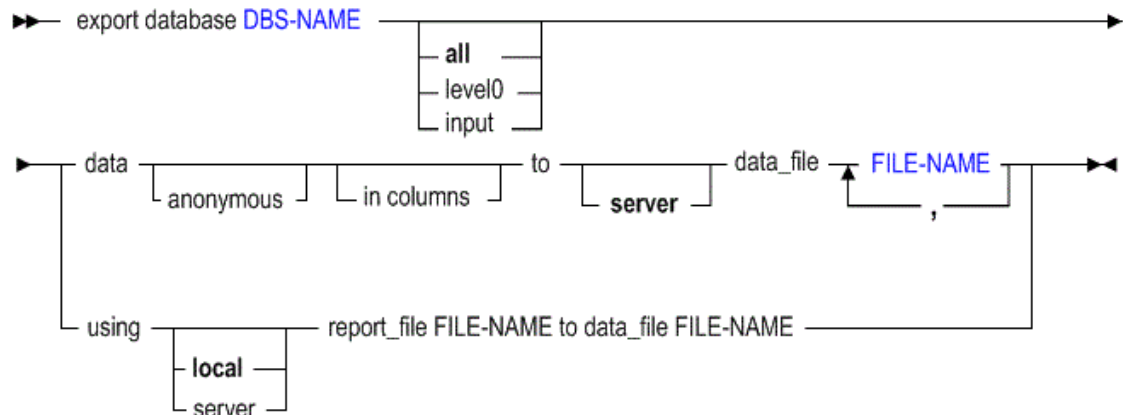
[Click here for aggregate storage version](#)

Using this statement, you can export all data, level-0 data, or input-level data, which does not include calculated values.

Essbase writes data export files to the cube directory, unless an alternate path is specified using FILEGOVPATH configuration. To use Report Writer, export the data using a report file. Export data files cannot be written to the client computer.

The minimum permission required to export data is Database Access.

Syntax



- [DBS-NAME](#)
- [FILE-NAME](#)

Keywords

You can export data from a database in the following ways using MaxL **export database**.



Note:

Exporting data does not clear the data from the database.

export database DBS-NAME [all] data ...

Export all data in the specified cube to the application/cube directory (or to whichever Essbase Server directory the administrator has specified for FILEGOVPATH).

Example:

```
export database Sample.Basic data to server data_file 'myfilesamp.txt';
```

export database DBS-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.

Example:

```
export database Sample.Basic level0 data to data_file 'sbleaf.txt';
```

export database DBS-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.

Example:

```
export database Sample.Basic input data to data_file 'sbinput.txt';
```

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.

Example:

```
export database Sample.Basic data in columns to data_file 'sbcols.txt';
```

export database DBS-NAME ... data anonymous ...

Export data in anonymized format. Anonymization removes the risk of sensitive data disclosure, and can be used in case sample data needs to be provided for technical support. Essbase replaces real data values with incremental values beginning with 0, increasing by 1 for each value in the block.

Example:

```
export database Sample.Basic data anonymous to data_file 'sbobsured.txt';
```

export database DBS-NAME ... using ... report_file ...

Run a stored report script, exporting a subset of the database. The export file is written to the current directory where you are logged in to MaxL.

Example:

```
export database Sample.Basic using server report_file 'link' to data_file  
'LINKREPORT.TXT';
```

Notes

- This statement requires the database to be started.
- To export data in parallel, specify a comma-separated list of export files, up to a maximum of 1024 file names. For example:

```
export database sample.basic data to data_file 'asdf_1.txt', 'asdf_2.txt';
```

The number of file names determines the number of export threads. The number of available block-address ranges limits the number of export threads that Essbase actually uses. Essbase divides the number of actual data blocks by the specified number of file names (export threads). If there are fewer actual data blocks than the specified number of export threads, the number of export threads that are created is based on the number of actual data blocks. For example, if the block storage database is very small, with only 100 data blocks, Essbase will use only 100 threads, even if you specify a higher number. This approach results in a more even distribution of data blocks between export threads.

 **Note:**

In specifying the number of export files, it is important to consider the number of available CPU cores and I/O bandwidth on the computer on which Essbase Server runs. Specifying too large a number can result in poor performance.

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 `exportfile.txt`, the next additional file is `exportfile_1.txt`.

- 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.
- MaxL cannot export databases with names containing hyphens (-).

Export LRO

The MaxL `export lro` statement helps you export linked-reporting-object information, and binary files if the Essbase cube has file-type LROs, to a directory. The minimum application permission required to export LROs is Database Access.

Syntax

```

▶▶ export database DBS-NAME lro to server directory DBS-EXPORT-DIR
local FULL-EXPORT-DIR ▶▶

```

- `DBS-NAME`
- `DBS-EXPORT-DIR`
- `FULL-EXPORT-DIR`

Keywords

You can export LRO information from a cube in the following ways using MaxL **export lro**.

export database DBS-NAME lro to server directory ...

Export the LRO information to the cube directory on the Essbase Server to which you are connected.

export database DBS-NAME lro to local directory ...

Export the LRO information to a local directory you specify.

Notes

- This statement requires the cube to be started.
- MaxL creates exactly one local 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 local export directory already exists, the export LRO statement fails. This is a safeguard against overwriting existing export directories.
- If you do not specify a *full path* for an export directory, MaxL uses your short directory specification (**DBS-EXPORT-DIR**) as a suffix, and creates the destination export-directory in the `<Application Directory>/app` directory on the Essbase Server, with a prefix of `appname-dbname-`. If you do specify a full path, MaxL creates whatever directory you specify.

If you do not know where *Application Directory* is in your environment, refer to Environment Locations in the Essbase Platform.

- Relative paths (for example, `'../exports/lros'`) are not supported.
- The export directory and `.exp` file are on the Essbase Server filesystem, but are not displayed in the Files catalog in Essbase web interface.
- When MaxL exports LROs from a cube, if the cube 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.
- The exported lros can be imported into a cube using [import lro](#).

Examples

```
export database sample.basic lro to server directory '/scratch/exports/lros';
```

Exports LRO-catalog information, and binary files if the cube has file-type LROs, to a server directory `lros`, under `/scratch/exports/`. The statement will fail unless `/scratch/exports/` already exists. The directory contains file-type LROs, if applicable, and the LRO-catalog export file `lros.exp`. These can be brought back into a cube using [import lro](#). The export directory and `.exp` file are visible on the Essbase server machine, but are not in the Files catalog in Essbase web interface.

The example will work only once, because MaxL will not overwrite the existing directory.

```
export database sample.basic lro to server directory 'exportedLROs';
```

Exports LRO-catalog information, and binary files if the cube has file-type LROs, to a server directory `<Application Directory>/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 cube using [import lro](#).

The example will work only once, because MaxL will not overwrite the existing directory.

```
export database sample.basic lro to local directory 'C:\\Oracle\\EssbaseMaxL\\myexports';
```

The example above exports LROs to a local directory on a Windows MaxL client. The statement succeeds only if directory C:\\Oracle\\EssbaseMaxL exists.

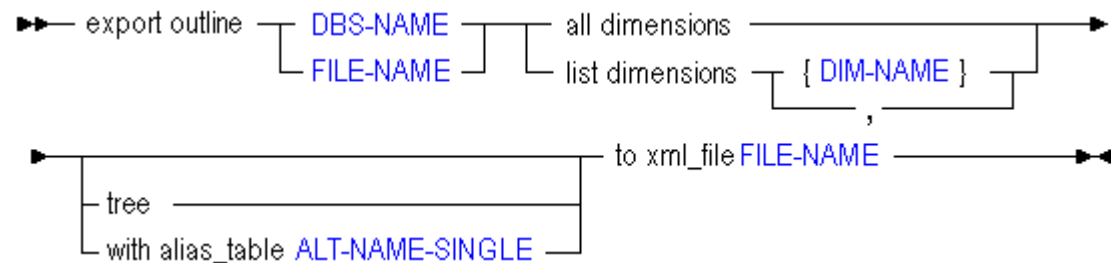
Export Outline

The MaxL export outline statement helps you export Essbase metadata, either from the active cube outline or an input outline file, to a specified XML file.

Export outline files must be written to a location on the Essbase Server or client machine on which the **export outline** MaxL statement is run.

The minimum application permission required to export the outline is Database Manager.

Syntax



- DBS-NAME
- FILE-NAME
- DIM-NAME
- ALT-NAME-SINGLE

Keywords

You can export metadata information from a cube in the following ways using MaxL **export outline**.

export outline DBS-NAME ...

Export the outline for the specified cube name.

export outline FILE-NAME ...

Export the outline with the specified .otl file name.

export outline ... all dimensions ...

Export information about all dimensions in the cube.

export outline ... list dimensions ...

Export information about only the listed dimensions. Specify each dimension name within curly braces, and separated by commas.

export outline ... tree ...

Export only the member names in the hierarchy, omitting full metadata details.

export outline ... with alias_table ALT-NAME-SINGLE ...

Export using only the member names indicated in the specified alias table.

export outline ... to xml_file FILE-NAME

Specify the absolute path to the output XML file, including the file name.

Notes

- This statement requires the cube 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
 - 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)

Examples

Linux Example 1

```
export outline sample.basic all dimensions to xml_file '/scratch/myexports/  
basic.xml';
```

Exports all outline information from Sample.Basic to the specified XML file, basic.xml.

Linux Example 2

```
export outline sample.basic list dimensions {"Product", "Market"} tree to  
xml_file '/scratch/myexports/basic.xml';
```

Exports information about Product and Market dimensions from Sample.Basic to the XML file.

Windows Example

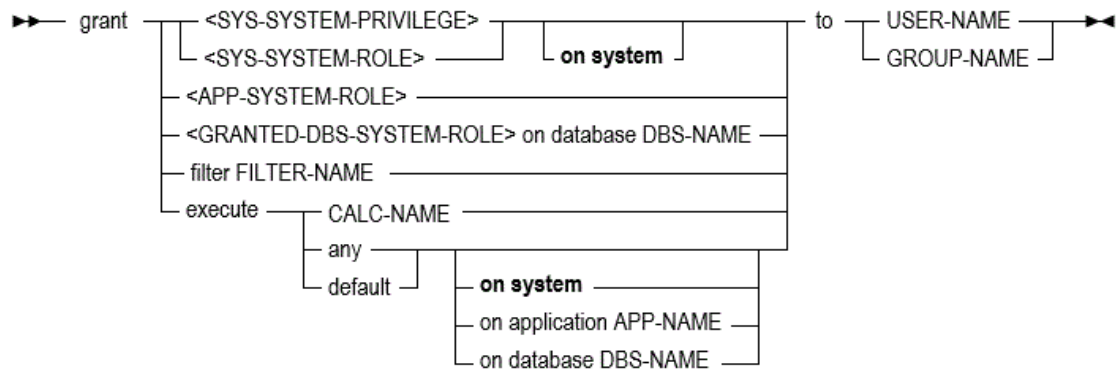
```
export outline sample.basic all dimensions with alias_table "Default" to  
xml_file 'C:\\myexports\\basic.xml';
```

Exports information about all dimensions in Sample.Basic, using only default alias names.

Grant

The MaxL grant statement helps you assign an Essbase security filter or a stored calculation to a user or group. If Essbase uses EPM Shared Services security mode, you can also grant privileges and roles.

Syntax



- [FILTER-NAME](#)
- [USER-NAME](#)
- [GROUP-NAME](#)
- [CALC-NAME](#)
- [System-Level System Privileges](#)
- [System-Level System Roles](#)
- [Application-Level System Roles](#)
- [Database-Level System Roles](#)

Keywords

You can grant permissions to users and groups in the following ways using MaxL **grant**.

grant SYS-SYSTEM-PRIVILEGE to ...

Grant a [system-level privilege](#) to a user or group. Available only if Essbase uses EPM Shared Services security mode.

Example:

```
grant create_application to user3;
```

grant SYS-SYSTEM-ROLE to ...

Grant a [system-level role](#) to a user or group. Available only if Essbase uses EPM Shared Services security mode.

Example:

```
grant no_access to user3;
```

grant APP-SYSTEM-ROLE to ...

Grant an [application-level role](#) to a user or group. Available only if Essbase uses EPM Shared Services security mode.

Example:

```
grant manager to user3;
```

grant GRANTED-DBS-SYSTEM-ROLE to ...

Grant a [database-level role](#) to a user or group. Available only if Essbase uses EPM Shared Services security mode. In this mode, user or group access can only be assigned at the

application level. If the application has more than one database (cube), trying to change access for one cube implicitly changes the access privilege on all cubes belonging to that application.

Example:

```
grant read to user3;
```

grant 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.

Example:

```
grant filter Sample.basic.filter8 to user5;
```

grant execute CALC-NAME to...

Grant the user or group permission to run the specified stored calculation script.

Example:

```
grant execute Sample.Basic.EBudg to user3;
```

grant execute any on system to...

Grant the user or group permission to run any calculation against any database on the Essbase Server.

Example:

```
grant execute any to calcmgr;
```

grant execute any on application...to...

Grant the user or group permission to run any calculation against any databases in the specified application.

Example:

```
grant execute any on application Sample to calcmgr;
```

grant execute any on database...to...

Grant the user or group permission to run any calculation against the specified database.

Example:

```
grant execute any on database Sample.Basic to calcmgr;
```

grant execute default on system to...

Grant the user or group permission to run the default calculation against any database on the Essbase Server.

Example:

```
grant execute default on system to calcmgr;
```

grant execute default on application...to...

Grant the user or group permission to run the default calculation against any databases in the specified application.

Example:

```
grant execute default on application Sample to appcalcmgr;
```

grant 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**.

Example:

```
grant execute default on database Sample.Basic to dbcalcmgr;
```

Notes

Granting Filters

Users may be granted multiple filters per database.

Granting Calculations

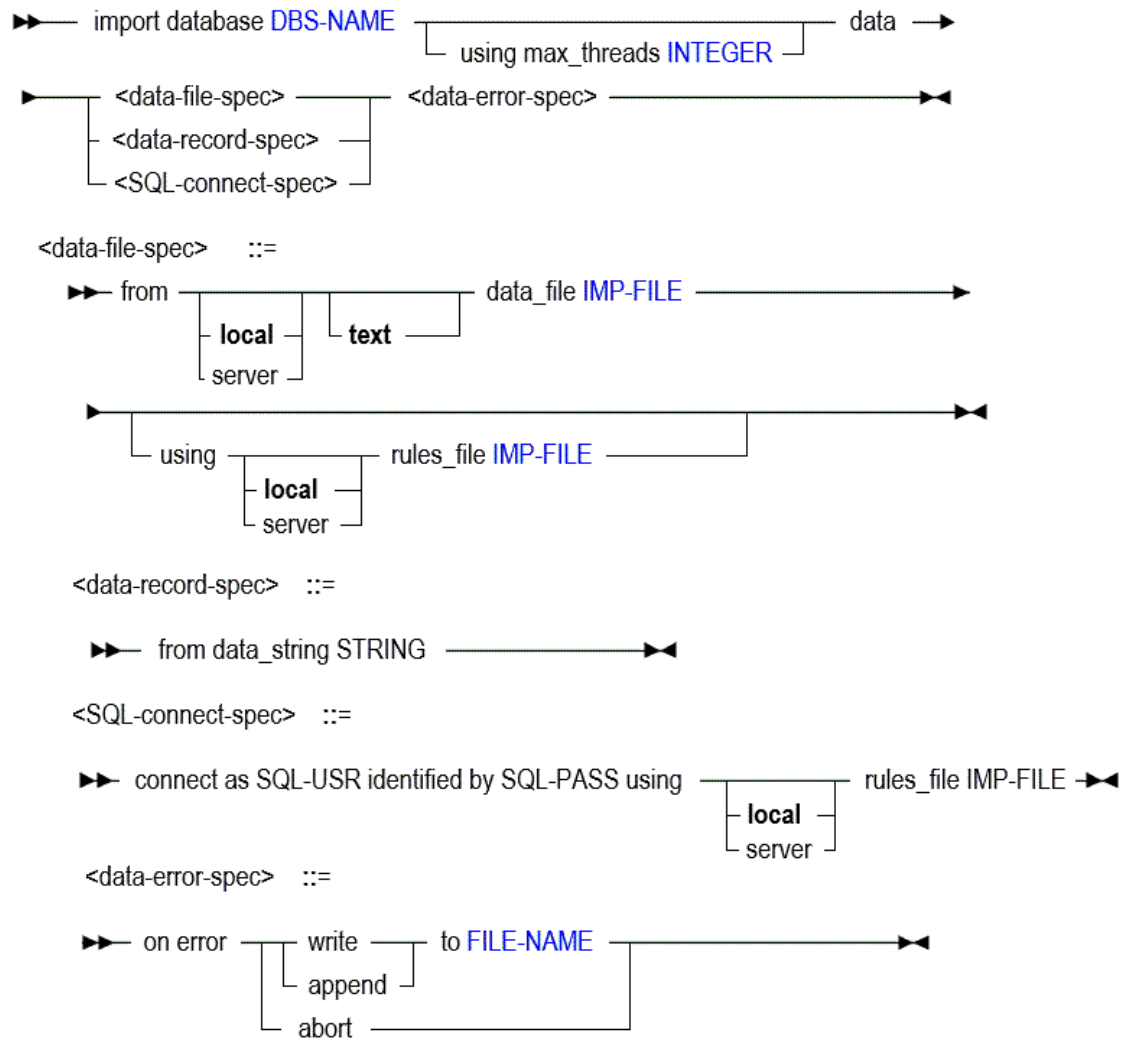
A user or group can run as many calculation scripts as needed per database. Therefore, granting a calculation adds it to the user or group's list of calculations. **Grant execute any** gives the user or group permission to execute all calculations, including the default calculation.

Import Data

The MaxL import data statement helps you load data into an Essbase database. You can import data from data files or external sources, with or without a rules file. The minimum application permission required to load data or dimensions is Database Update.

[Click here for aggregate storage version](#)

Syntax



- DBS-NAME
- INTEGER
- IMP-FILE
- FILE-NAME

Keywords

You can import data to a database in the following ways using MaxL **import data**.

import database DBS-NAME data from ...

Specify whether the data import file(s) are local or on the server, and specify the type of import file(s).

Example:

```
import database Sample.Basic data from server data_file 'Data_Basic.txt'
using server rules_file 'Data' on error write to 'sbdataload.err';
```

To import from multiple files in parallel, use the wildcard characters * and/or ? in the IMP-FILE name so that all intended import files are matched.

- * substitutes any number of characters, and can be used anywhere in the pattern. For example, `day*.txt` matches an entire set of import files ranging from `day1.txt` - `day9.txt`.
- ?* substitutes one occurrence of any character, and can be used anywhere in the pattern. For example, `0?*-2011.txt` matches data source files named by date, for the single-digit months (Jan to Sept).

Example:

```
import database Sample.Basic data from server data_file 'asdf*.txt' using
server rules_file 'hjkl' on error write to 'asdf_load.err';
```

import database DBS-NAME using max_threads INTEGER ...

For parallel data loads, optionally specify a maximum number of threads to use. If this clause is omitted for a parallel data load, Essbase uses a number of pipelines equal to the lesser of number of files, or half the number of CPU cores.

Example:

```
import database Sample.Basic using max_threads 5 data from server data_file
'asdf*.txt' on error append to 'asdf_load.err';
```

import database ... using ... rules_file

Import data into the database using a specified load rule. A separate rule file is required for each unique, non-Essbase source of data. If you're only re-importing data from native Essbase export files, you may not need to use a rule. If you are using a load rule for a parallel data load, all the data files must be able to use the same one.

Example:

```
import database Sample.Basic data from server data_file 'Data_Basic.txt'
using server rules_file 'Data' on error write to 'basic_load.err';
```

import database ... data ... 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. See any Example on this page for usage.

import database ... data ... from data_string

Load a single data record into the selected database. The string following `data_string` must be a contiguous line, without newline characters.

Example:

```
import database sample.basic data from data_string '"Sales" "100-10" "New
York" "Jan" "Actual" 678' on error abort;
```

import database ... data ... connect as SQL-USR ...

If you are importing from an SQL source, you must always use a rule file. Provide the appropriate user name and password:

- If the network connectivity to the source data is saved in an Essbase connection and Datasource, provide your Essbase credentials in the MaxL statement. For example:

```
import database Sample.Basic data connect as "Essbaseadmin" identified by
"Essbasepa55w0RD" using server rules_file "myrulefile" on error write to
'loadds.err';
```

- Otherwise, provide the user name and password required to connect to the external RDBMS source. For example:

```
import database Sample.Basic data connect as "RDBMSuser" identified by
"RDBMSpa55w0RD" using server rules_file "myrulefile" on error write to
'loadds.err';
```

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.

Catalog Path Example

When the Essbase file catalog location is unspecified, the cube directory is the assumed location.

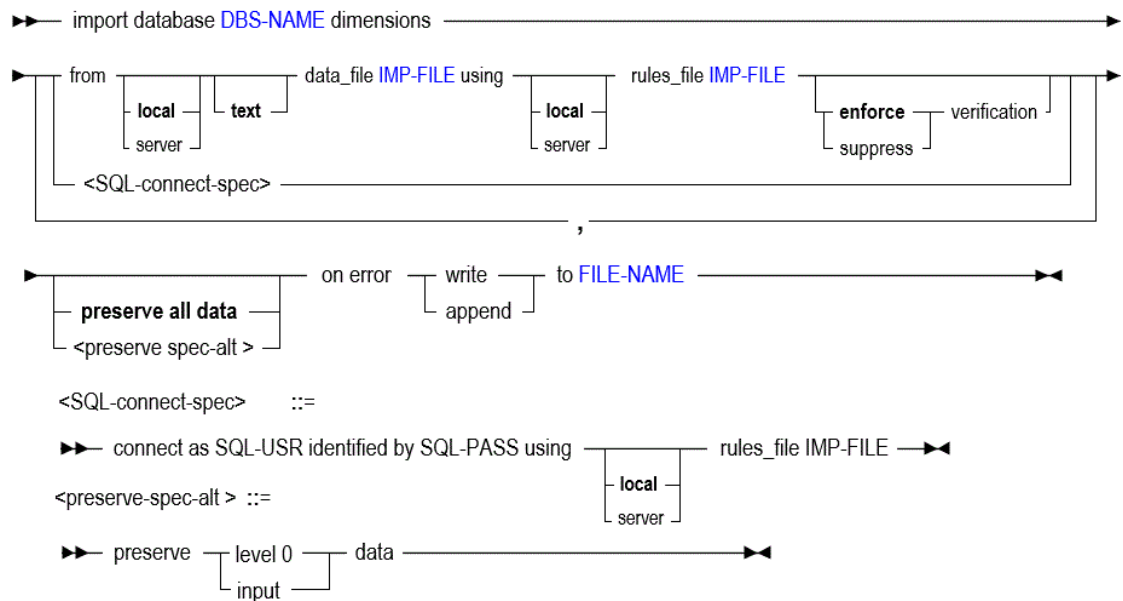
The following example performs a data load using a data file stored in the shared folder of the Essbase file catalog. The rule file is in the cube directory for Sample Basic.

```
import database 'Sample'. 'Basic' data from server data_file 'catalog/shared/
Data_Basic' using server rules_file 'Data' on error abort;
```

Import Dimensions

The MaxL import dimensions statement helps you load dimensions into an Essbase database. The minimum application permission required to load data or dimensions is Database Update.

Syntax



- [DBS-NAME](#)
- [IMP-FILE](#)
- [FILE-NAME](#)

Keywords

You can import dimensions to a database in the following ways using MaxL **import dimensions**.

import database DBS-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.

import database ... dimensions ... using ... rules_file ...

Import dimensions into the database outline using a specified rules file.

import database ... dimensions ... enforce verification ...

Verify the outline resulting from the dimension build. This is the default behavior.

import database ... dimensions ... suppress verification ...

Do not verify the outline resulting from the dimension build.

Caution:

Using this option defers restructuring.

import database ... dimensions ... preserve all data ...

If you need to preserve all data when importing dimensions, specify **preserve all data**.

import database ... dimensions ... preserve level 0 data ...

Preserve only level-0 when importing dimensions.

import database ...dimensions ... preserve input data ...

Preserve only input data when importing dimensions.

import database ... dimensions ... on error ...

Required. Tell Essbase what to do in case of errors during the dimension build: abort the operation, or write or append to a specified error log. See any Example on this page for usage.

import database ... dimensions ... connect as SQL-USR ...

If you are importing from an SQL source, you must always use a rule file. Provide the appropriate user name and password:

- If the network connectivity to the source data is saved in an Essbase connection and Datasource, provide your Essbase credentials in the MaxL statement. For example:

```
import database Sample.Basic dimensions connect as "Essbaseadmin"
identified by "Essbasepa55w0RD" using server rules_file "myrulefile" on
error write to 'mydimbuild.err';
```

- Otherwise, provide the user name and password required to connect to the external RDBMS source.

```
import database Sample.Basic dimensions connect as "RDBMSuser" identified
by "RDBMSpa55w0RD" using server rules_file "myrulefile" on error write to
'mydimbuild.err';
```

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.**

Examples

The following example performs a dimension build using a data file stored in the shared folder of the Essbase file catalog. The rule file is in the cube directory for Sample Basic.

```
import database 'Sample'. 'Basic' dimensions from server data_file 'catalog/
shared/addproducts' using server rules_file 'addproduct' on error write to
"dimrule.err";
```

The following example performs a dimension build using a data file and rule file located in the cube directory for Sample Basic. When you specify a server data file and the Essbase file catalog location is unspecified, as in this example, the cube directory is assumed to be the

location of the files. In other words, if `dims.txt` and `rulesfile.rul` both exist in the cube directory, then the dimension build will use these files.

```
import database Sample.Basic dimensions from server data_file 'dims.txt'
using rules_file 'rulesfile.rul' on error append to "dimbuild.log";
```

The following example performs a dimension build using a data file stored in a user directory in the Essbase file catalog. The rule file is in the shared directory in the catalog.

```
import database 'Sample'. 'Basic' dimensions from server data_file 'catalog/
users/user1/Dim_Market' using server rules_file 'catalog/shared/Dim_Market'
on error write to "dimrule.err";
```

Import LRO

The MaxL `import lro` statement helps you import Linked Reporting Objects (LROs) into an Essbase cube.

You import 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).

Syntax

```
▶▶ import database DBS-NAME lro from  directory IMPORT-DIR ◀◀
```

- `DBS-NAME`
- `IMPORT-DIR`

Keywords

You can import exported LRO information to a cube using MaxL **import lro** in the following ways. The minimum application permission required for this action is Database Update.

import database DBS-NAME lro from local directory IMPORT-DIR

Import Linked Reporting Objects (LROs) from the specified export directory on the local machine.

import database DBS-NAME lro from server directory IMPORT-DIR

Import Linked Reporting Objects (LROs) from the specified export directory on Essbase Server.

Notes

- This statement requires the cube to be started.
- The specified LRO 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](#)) in the path name.

Windows Example

Referencing an absolute path to an export directory:

```
import database sample.basic lro from local directory 'C:\\Oracle\\  
\\EssbaseMaxL\\myexports';
```

Linux Example

Referencing an absolute path to an export directory:

```
import database sample.basic lro from local directory '/scratch/exports/lros';
```

Generic Example

The following example works on Windows or Linux. It references a server directory, *<Application Directory>*/app/Sample-Basic-exportedLROs:

```
import database sample.basic lro from server directory 'Sample-Basic-  
exportedLROs';
```

If you do not know where *Application Directory* is in your environment, refer to Environment Locations in the Essbase Platform.

Query Application

The MaxL query application statement for block storage (BSO) applications helps you get the maximum cache size of an Essbase application. To issue this statement, the application must be started first, and you need at least Database Access permission for the application.

[Click here for aggregate storage version](#)

Syntax

```
▶▶ query application APP-NAME get cache_size ───────────▶▶
```

APP-NAME

Keywords

You can query the application state information in the following ways using MaxL **query application**.

query application APP-NAME get cache_size

Check the current maximum size setting to which the application cache may grow. The application cache grows dynamically until it reaches this limit. The application cache can help you manage memory usage for retrievals. For block storage cubes, the application cache is used for hybrid aggregation.

Example

```
alter system load application ASOSamp;  
query application ASOSamp get cache_size;
```

Query Archive_File

The MaxL **query archive_file** statement helps you retrieve information about the Essbase database backup archive file. The archive file is created using **alter database DBS-NAME begin | end archive**.

Syntax

```
query archive_file FILE-NAME {  
  get overview  
  list disk volume  
}
```

FILE-NAME

Keywords

You can query archive file information in the following ways using MaxL **query archive_file**. To issue this statement, the database must be started first, and you need at least Database Access permission for the application.

query archive_file ... get overview

Retrieve the application name, database name, and time of archive.

query archive_file ... list disk volume

Retrieve a list of disk volume names.

Example

```
query archive_file 'samplebasic.arc' get overview;
```

Retrieves overview information about the `samplebasic.arc` backup archive file.

```
query archive_file 'samplebasic.arc' list disk volume;
```

Retrieves disk volume information about the `samplebasic.arc` backup archive file.

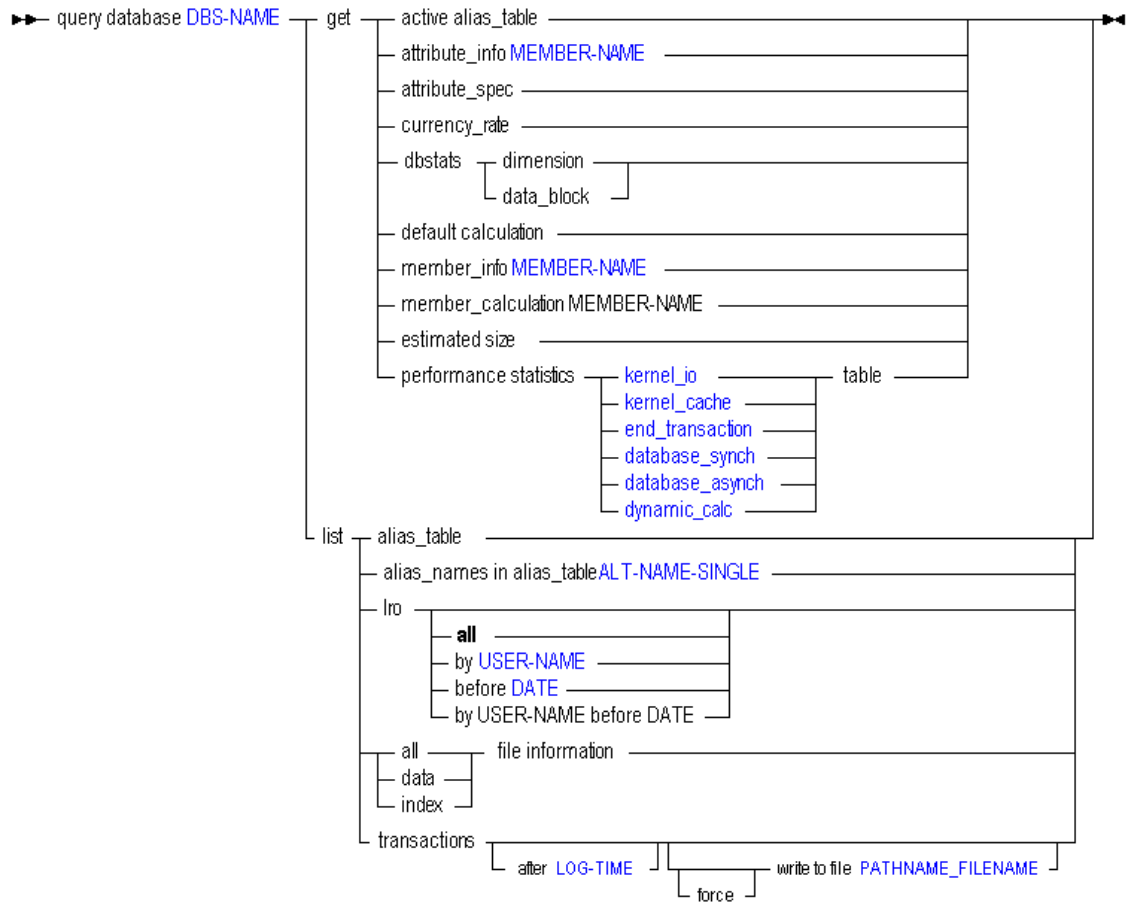
Query Database

The MaxL **query database** statement helps you retrieve advanced information about the current state of an Essbase database that is running.

[Click here for aggregate storage version](#)

This statement requires the database to be started.

Syntax



- DBS-NAME
- MEMBER-NAME
- ALT-NAME-SINGLE
- USER-NAME
- DATE
- LOG-TIME
- PATHNAME_FILENAME

Keywords

You can query for database information in the following ways using MaxL **query database**. The minimum application permission required for most **query database** actions is Database Access, with exceptions noted.

query database DBS-NAME get active alias_table

Display the active alias table for the user issuing the statement.

Example:

```
query database Sample.Basic get active alias_table;
```

query database DBS-NAME get attribute_info MEMBER-NAME

Get attribute member, dimension, and name information for the specified attribute member.

Example:

```
query database Sample.Basic get attribute_info 'Ounces';
```

query database DBS-NAME 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.

Example:

```
query database Sample.Basic get attribute_spec;
```

query database DBS-NAME get currency_rate

Display the currency rate for every currency partition. The database must be started.

Example:

```
query database Sample_Currency.Internl get currency_rate;
```

query database DBS-NAME get dbstats dimension

Get information about dimensions.

Output

The **index_type** field values are numeric, and translate as follows:

```
0   Dense
1           Sparse
3           None (database is aggregate storage)
```

query database DBS-NAME get dbstats data_block

Get information about data blocks. The information returned has little relevance to aggregate storage databases.

Output

The type field values are numeric, and translate as follows:

```
0           Array
1           AVL (or "B+ Tree")
```

query database DBS-NAME get default calculation

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.

Example:

```
query database Sample.Basic get default calculation;
```

query database DBS-NAME get member_info MEMBER-NAME

Get information on a specific member.

Example:

```
query database Sample_Dynamic.Basic get member_info 'Profit per Ounce';
```

Output Fields for Member Info Query in MaxL

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

query database DBS-NAME get member_calculation MEMBER-NAME

View the formula associated with the selected member.

Example:

```
query database sample.basic get member_calculation 'Profit per Ounce';
```

Displays the formula associated with the 'Profit per Ounce' member.

query database DBS-NAME get estimated size

Get an estimate of the cube size; that is, display an estimate of the number of blocks a database will create after full calculation (CALC ALL). The estimate is 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).
Example:

```
query database Sample.Basic get estimated size;
```

query database DBS-NAME get performance statistics TABLE-TYPE table

Display one of several choices of [performance statistics tables](#) returning detailed information about the block storage cube statistics. Before you can use this statement, you must enable performance statistics gathering, using [alter database DBS-NAME set performance statistics enabled](#).

Example:

```
alter database Sample.Basic set performance statistics enabled;
```

```
query database Sample.Basic get performance statistics dynamic_calc table;
```

query database DBS-NAME list alias_table

Get a list of alias tables that are defined for the database.

Example:

```
query database Sample.Basic list alias_table;
```

query database DBS-NAME 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.

Example:

```
query database Sample.Basic list alias_names in alias_table 'Long Names';
```

query database DBS-NAME 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.

Example:

```
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.

query database DBS-NAME list <all | data | index> 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.

Example:

```
query database Sample.Basic list all file information;
```

query database DBS-NAME 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).

Example:

```
query database Sample.Basic list transactions;
```

To use this statement, you must have enabled transaction logging using **alter database**.

 **Note:**

Transaction logging and replay is available only for backward compatibility support in Essbase 21c. Features added after Essbase 11g On-Premise are not supported for transaction logging and replay; including (but not limited to):

- Batch outline editing
- Application workbooks and Cube Designer activity
- Scenario management
- External data load using a Datasource
- REST API data loads and other updates
- Federated partitions
- MDX inserts
- Drill through transactions

query database DBS-NAME list transactions after LOG-TIME

Display database transactions that were logged after the specified time. Enclose the TIME value in quotation marks; for example: '11_20_2007:12:20:00'

**query database DBS-NAME 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 file name is provided, Essbase writes the file to the application directory.

When writing to an output file that already exists, you must use the **force** grammar to overwrite the file.

Example:

```
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';
```

query database DBS-NAME list transactions force write to file PATHNAME_FILENAME
Overwrite the contents of an existing output file.

**query database DBS-NAME 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.

Refresh Custom Definitions

The MaxL refresh custom definitions statement helps you update the record of any custom-defined Essbase calculation functions and macros that are associated with an application, without restarting the application.

Syntax

►► refresh custom definitions on application **APP-NAME** ◄◄

APP-NAME

Keywords**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.

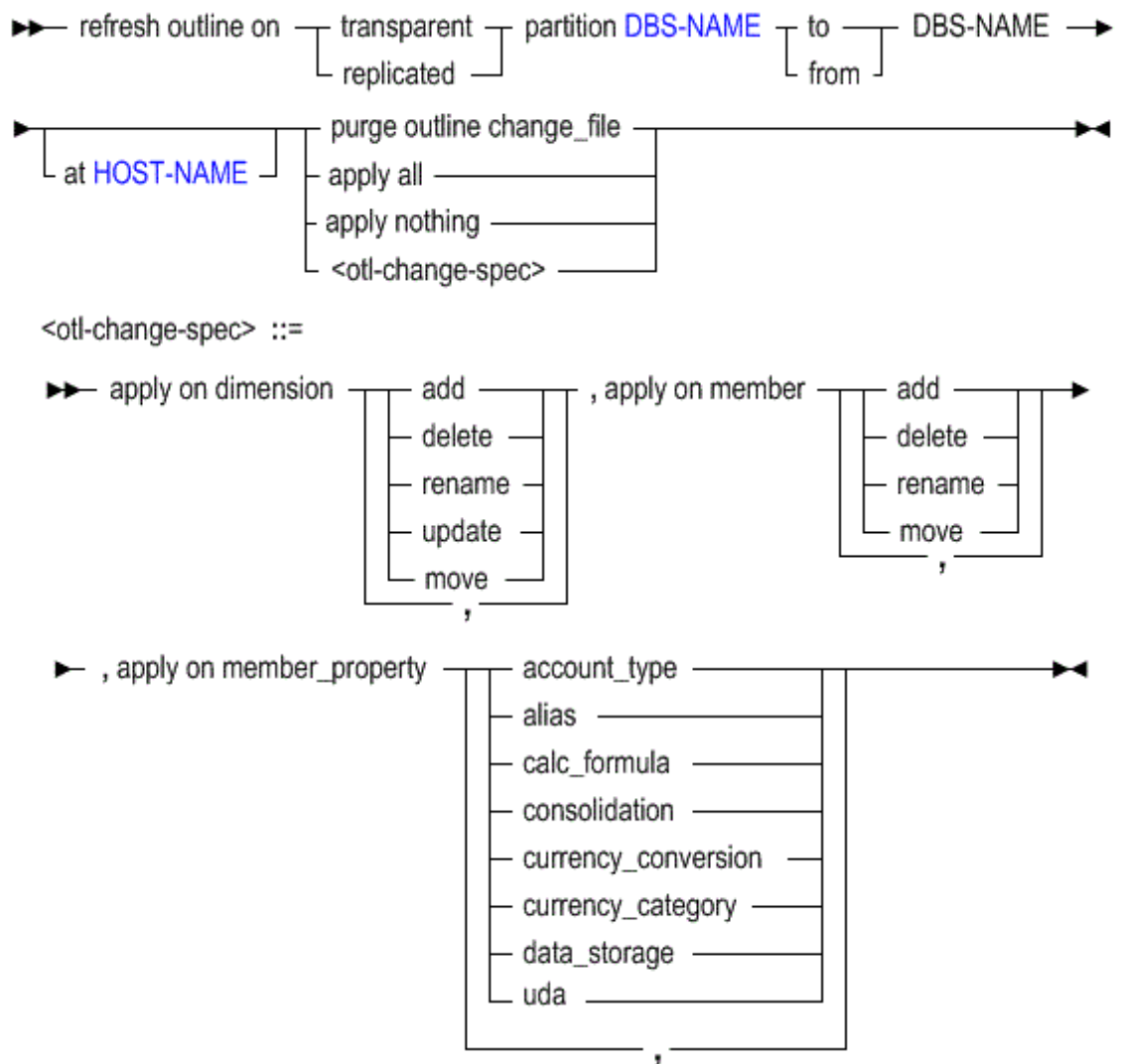
Refresh Outline

The MaxL refresh outline statement helps you synchronize the outlines between partitioned Essbase databases.

Use this statement in the event that one Essbase 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 enabled for partitions that involve aggregate storage databases.

Syntax



- [DBS-NAME](#)
- [HOST-NAME](#)

You can synchronize the outlines between partitioned databases using **refresh outline**.

Keywords

...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.

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...

Refresh the target outline with all or some dimension changes made to the source outline.

- **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.

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: `apply on dimension add, move`.

apply on member...

Refresh the target outline with all or some physical member changes made to the source outline. Requires **apply on dimension update**.

- **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.

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: `apply on dimension update, apply on member add, move`.

apply on member_property...

Refresh the target outline with all or some member property changes made to the source outline. Requires **apply on dimension update**.

- **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.

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: `apply on dimension update, apply on member_property calc_formula`.

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).

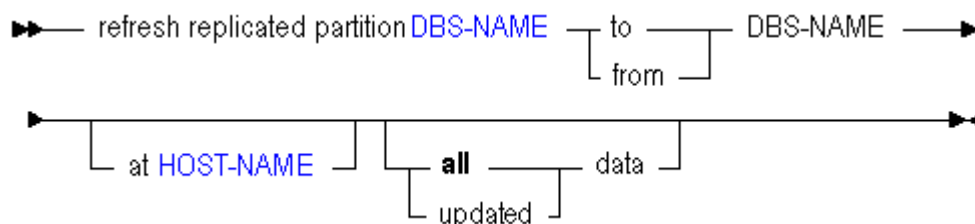
```
refresh outline on replicated partition Sampeast.east to Samppart.company
  apply on dimension update, apply on member rename, apply on member_property
  account_type;
```

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

The MaxL refresh replicated partition statement helps you synchronize the current Essbase replicated-partition target cube from the remote (second DBS-NAME) source partition. Database Manager permission for each cube is required.

Syntax



- [DBS-NAME](#)
- [HOST-NAME](#)

You can update a replicated-partition using **refresh replicated partition**.

Keywords

...to...

Use the current replicated-partition source cube to refresh the remote target partition.

...from...

Refresh the current replicated-partition target cube from the remote source partition.

...updated data

Refresh a replicated-partition cube only with data that has been updated since the last refresh.

...all data

Refresh a replicated-partition cube with all data, regardless of the last refresh.

Example

```
refresh replicated partition sampeast.east to samppart.company at "https://  
myEssbase-myDomain.analytics.us2.example.com/essbase/agent" all data;
```

Performance Statistics in MaxL

MaxL statements can help you collect and analyze Essbase performance statistics. Performance statistics help you understand whether the databases are in good running condition or need modifications to improve performance.

About Performance Statistics

[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 <dbs-name> set performance statistics enabled](#). You also use `alter database` to return to zero the statistical *persistence* (length) and *scope* (granularity).

The output of `query database` can vary depending on what the system has just done, how long statistics have been gathered, and the persistence level 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.

Kernel Input/Output Statistics

The Kernel I/O Statistics table summarizes input/output for the entire application.

There is one Kernel I/O Statistics table per application.

Persistence/Scope of this table: **med/server**

Table 3-4 Kernel IO Statistics

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 you in determining how to size Essbase 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 Essbase kernel uses 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.

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**

Table 3-5 Kernel Cache Statistics

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.
# 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 or replacements is low or zero, the cache might be set too large.

Table 3-5 (Cont.) Kernel Cache Statistics

Kernel Cache Statistic	Description
# 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.

This table shows the cleanup state at the end of a transaction. Because the DBWriter only operates during idle times, measuring its activity can give you 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**

Table 3-6 DB Sync IO Statistics

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.

Table 3-6 (Cont.) DB Sync IO Statistics

DataBase Sync I/O	Count	Bytes	Ttotal (ms)	Tave (ms)
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**

Table 3-7 DB Async IO Statistics

DataBase Async 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.

Table 3-7 (Cont.) DB Async IO Statistics

DataBase Async I/O	Count	Bytes	Ttotal (ms)	Tave (ms)	Twait (ms)
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 (including DYNCALCCACHEMAXSIZE).

Table 3-8 Dynamic Calc Cache Statistics

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.

Table 3-8 (Cont.) Dynamic Calc Cache Statistics

Dynamic Calc Cache Statistic	Description
DCC Blks Waited	The number of dynamic calculator blocks that the system had to wait for.
DCC Blks Timeout	The number of times that Essbase timed out waiting for free space in the dynamic calculator cache.
DCC Max ThdQLen	The highest number of threads that were left waiting in queue for Dynamic Calc cache memory to be freed.

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;

```

MaxL Statements (Aggregate Storage)

Some MaxL statements are available specifically for Essbase applications and cubes that are in aggregate storage mode.

[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 object
- alter partition
- alter system
- alter tablespace
- alter trigger
- create application
- create database
- create filter
- create outline
- create partition
- create after-update trigger
- display application
- display calculation
- display database
- display filter
- display filter row
- display lock
- display object
- display partition
- display privilege
- display session
- display system
- display tablespace
- display trigger
- display variable
- drop application
- drop calculation
- drop database
- drop filter
- drop lock
- drop object
- drop partition
- drop trigger
- execute aggregate process
- execute aggregate build
- execute aggregate selection
- execute allocation

- [export data](#)
- [export query_tracking](#)
- [grant](#)
- [import data](#)
- [import dimensions](#)
- [import query_tracking](#)
- [login](#)
- [logout](#) (see [MaxL Shell Commands](#))
- [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. 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](#).



Note:

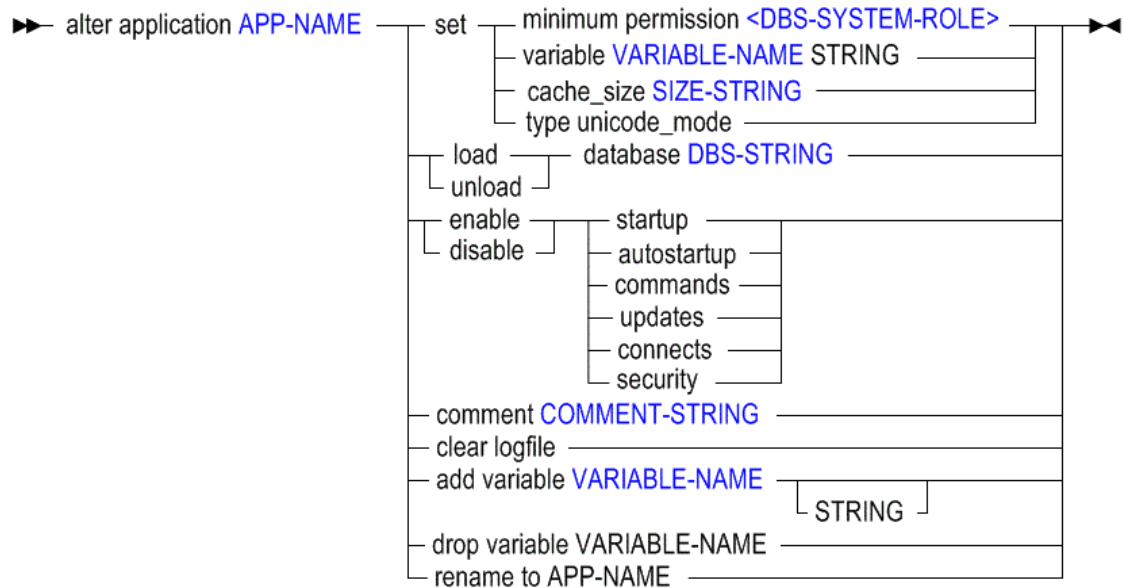
[Login](#) is part of the separate command shell grammar, not the MaxL language itself.

Alter Application (Aggregate Storage)

The MaxL **alter application** statement for ASO mode helps you change Essbase application-wide settings.

[Click here for non-aggregate storage version](#)

Syntax



- APP-NAME
- Database-Level System Roles
- VARIABLE-NAME
- SIZE-STRING
- DBS-STRING
- COMMENT-STRING
- VARIABLE-NAME

Keywords

Use MaxL **alter application** to change the following application-wide settings. The minimum application permission required for most of the statements is Application Manager, with exceptions noted.

alter application APP-NAME 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.

The options are `no_access`, `read`, `write`, `execute`, and `manager`.

Example:

```
alter application ASOsamp set minimum permission no_access;
```

alter application APP-NAME 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.

Example:

```
alter application ASOsamp set variable CurrMonth Oct;
```

alter application APP-NAME 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:

```
query application APP-NAME get cache_size;
```

If the ASODEFAULTCACHESIZE configuration setting is in use, either at application or server level, this MaxL statement has no effect.

Example:

```
alter application ASOsamp set cache_size 64MB;
```

Sets the maximum size of the aggregate storage cache to 64 MB.

alter application APP-NAME set type unicode_mode

Migrate an application to Unicode mode. Migration to Unicode mode cannot be reversed.

Example:

```
alter application ASOsamp set type unicode_mode;
```

alter application APP-NAME load database ...

Start (by loading into memory) an idle database. You need at least Database Access permission provisioned in the application.

Example:

```
alter application ASOSamp load database Basic;
```

alter application APP-NAME unload database ...

Stop (by unloading from memory) an active database. You need at least Database Access permission provisioned in the application.

Example:

```
alter application ASOSamp unload database Basic;
```

alter application APP-NAME enable startup

Permit all users who have at least Database Access permission to load (start) the application. Startup is enabled by default.

Example:

```
alter application ASOSamp enable startup;
```

alter application APP-NAME disable startup

Prevent all users from loading (starting) the application. Startup is enabled by default.

Example:

```
alter application ASOSamp disable startup;
```

alter application APP-NAME enable autostartup

Start the application automatically when Essbase Server starts. By default, autostartup is disabled.

Example:

```
alter application ASOSamp enable autostartup;
```

alter application APP-NAME disable autostartup

Do not start the application automatically when Essbase Server starts. By default, autostartup is disabled.

Example:

```
alter application ASOSamp disable autostartup;
```

alter application APP-NAME 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.

Example:

```
alter application ASOSamp enable commands;
```

alter application APP-NAME 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.

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.

By default, commands are enabled.

Example:

```
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 APP-NAME 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.

Example:

```
alter application ASOSamp enable updates;
```

alter application APP-NAME 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.

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.

Example:

```
alter application ASOSamp disable updates;
```

alter application APP-NAME 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.

Example:

```
alter application ASOSamp enable connects;
```

alter application APP-NAME 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. 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.

Example:

```
alter application ASOSamp disable connects;
```

alter application APP-NAME 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.

Example:

```
alter application ASOSamp enable security;
```

alter application APP-NAME 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.

Example:

```
alter application ASOSamp disable security;
```

alter application APP-NAME comment ...

Enter an application description (optional). The description can contain up to 80 characters.

Example:

```
alter application ASOSamp comment 'Sales application';
```

alter application APP-NAME clear logfile

Delete the application log located in the application directory. A new log is created for entries recording subsequent application activity.

Example:

```
alter application ASOSamp clear logfile;
```

alter application APP-NAME 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.

Example:

```
alter application ASOSamp add variable Month Nov;
```

alter application APP-NAME drop variable ...

Remove a substitution variable and its corresponding value from the application.

Example:

```
alter application ASOSamp drop variable Month;
```

alter application APP-NAME rename to

Rename the application. When you rename an application, the application and the application directory are renamed.

Example:

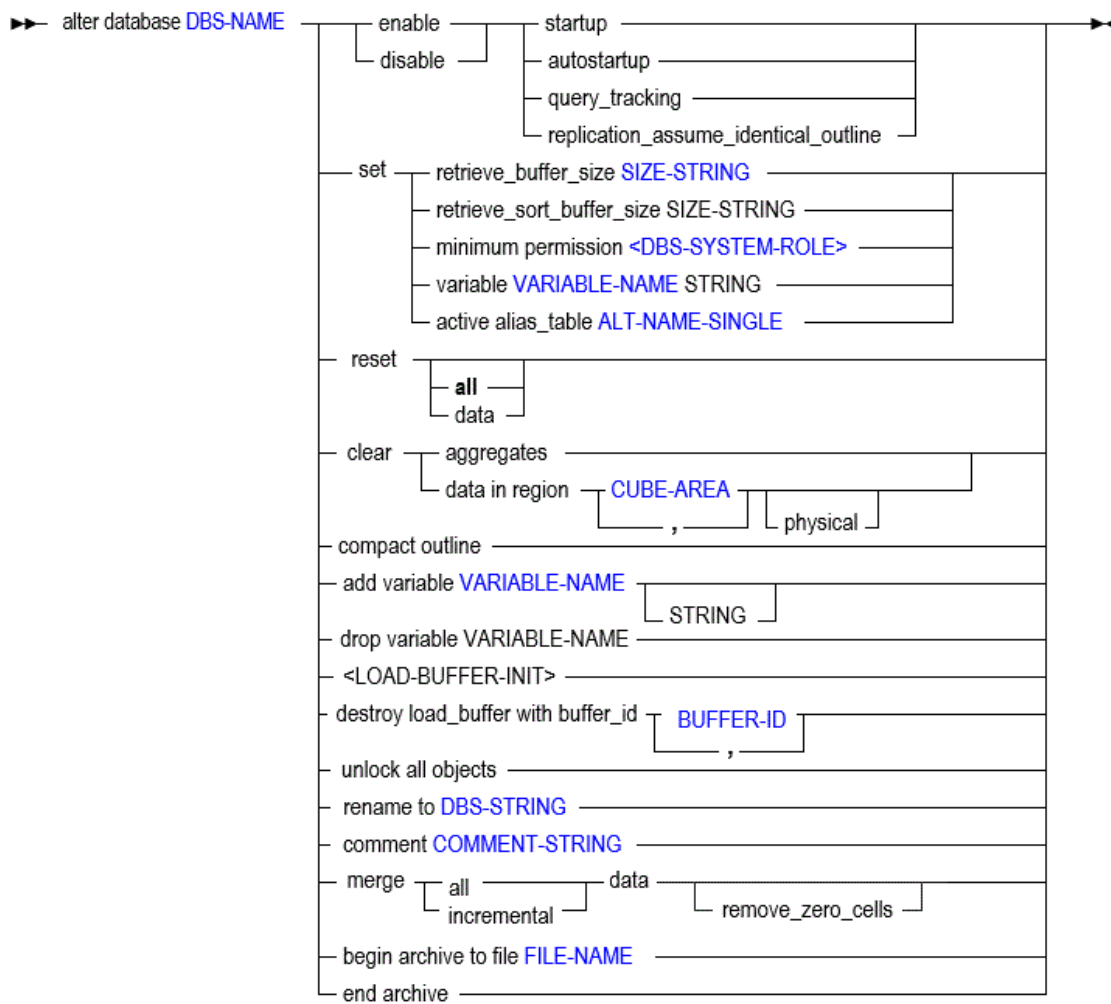
```
alter application ASOSamp rename to Sample2;
```

Alter Database (Aggregate Storage)

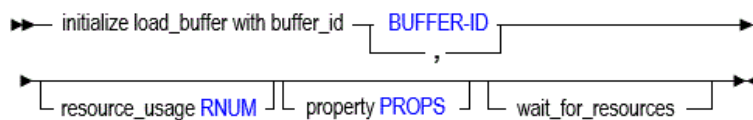
The MaxL alter database statement for ASO mode helps you change Essbase database-wide settings.

[Click here for non-aggregate storage version](#)

Syntax



<LOAD-BUFFER-INIT> ::=



- DBS-NAME
- DBS-SYSTEM-ROLE
- SIZE-STRING
- VARIABLE-NAME
- ALT-NAME-SINGLE
- CUBE-AREA
- BUFFER-ID
- RNUM
- PROPS
- DBS-STRING
- COMMENT-STRING

- [FILE-NAME](#)

Keywords

You can change aggregate storage (ASO) database settings in the following ways using MaxL **alter database**. The minimum application permission required for most of the statements is Database Manager, with exceptions noted.

alter database DBS-NAME 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.

Example:

```
alter database ASOSamp.Basic enable startup;
```

alter database DBS-NAME 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.

Example:

```
alter database ASOSamp.Basic disable startup;
```

alter database DBS-NAME 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.

Example:

```
alter database ASOSamp.Basic enable autostartup;
```

alter database DBS-NAME disable autostartup

Prevent automatic starting of the database when the application to which it belongs starts. Autostartup is enabled by default.

Example:

```
alter database ASOSamp.Basic disable autostartup;
```

alter database DBS-NAME enable query_tracking

Begin collecting query data for this database, to be used for query-based view optimization. To utilize the results of query tracking, use the optional based on query_data grammar in any of the following statements:

- [query database <db-name> list existing_views](#)
- [execute aggregate process](#)
- [execute aggregate selection](#)

Query tracking is on by default. To verify that it's enabled, use `query database appname.dbname get cube_size_info`.

 **Note:**

Query tracking and query tracing are different. Query *tracking* enables you to capture user retrieval statistics against an aggregate storage cube, so that Essbase can make view-based optimizations to improve the performance of aggregations. It is on by default. Related MaxL statements include:

```
import query_tracking
export query_tracking
alter database enable query_tracking
query database appname.dbname get cube_size_info
```

Query *tracing* helps you monitor Essbase query performance metrics for block storage cubes (including hybrid mode). It is off by default. If you enable it, Essbase logs metrics in a trace report. Related configuration parameters: TRACE_REPORT, QUERYTRACE, QUERYTRACETHRESHOLD, LONGQUERYTIMETHRESHOLD.

Example:

```
alter database ASOSamp.Basic enable query_tracking;
```

alter database DBS-NAME disable query_tracking

Stop collecting query data for query-based view optimization. Query tracking is on by default.

Example:

```
alter database ASOSamp.Basic disable query_tracking;
```

Turns off the harvesting of query data for the ASOsamp.Basic database.

alter database DBS-NAME 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 20 KB. The minimum size is 2 KB. Increasing the size may improve retrieval performance.

Example:

```
alter database ASOSamp.Basic set retrieve_buffer_size 20kb;
```

alter database DBS-NAME set retrieve_sort_buffer_size ...

Change the database retrieval sort buffer size. This buffer holds data until it is sorted. The default size is 20 KB. The minimum size is 2 KB. Increasing the size may improve retrieval performance.

Example:

```
alter database ASOSamp.Basic set retrieve_sort_buffer_size 20kb;
```


alter database DBS-NAME 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. This statement is supported for use in EPM Shared Services security mode only.

Example:

```
alter database ASOSamp.Basic set minimum permission no_access;
```

alter database DBS-NAME 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 (&).

Example:

```
alter database ASOSamp.Basic set variable CurrMonth 'Oct';
```

alter database DBS-NAME 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.

Example:

```
alter database ASOSamp.Basic set active alias_table 'Default';
```

To check which alias table is active, use **query database**.

alter database DBS-NAME reset

Clear all data and linked-reporting objects from the database, but preserve the outline.

 **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.

Example:

```
alter database ASOSamp.Basic reset;
```

alter database DBS-NAME reset all

Clear all data, Linked Reporting Objects, and the outline.

Example:

```
alter database ASOSamp.Basic reset all;
```

alter database DBS-NAME reset data

Same as using reset.

Example:

```
alter database ASOSamp.Basic reset data;
```

alter database DBS-NAME clear aggregates

Delete all aggregate views.

Example:

```
alter database ASOSamp.Basic clear aggregates;
```

Deletes all aggregate views in the ASOSamp.Basic database.

alter database DBS-NAME 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.

For a physical clear, use the MaxL statement with the **physical** keyword:

```
alter database appname.dbname clear data in region 'MDX set expression'  
physical;
```

To save time, you can use a comma-separated list of *MDX set expressions* to clear from multiple physical regions.

```
alter database ASOSamp.Basic clear data in region  
'{CrossJoin({[Promotions].[Coupon]},{[Time].[1st Half]}),  
  CrossJoin({[Promotions].[Coupon]},{[Time].[2nd Half]}}}'  
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.

For a logical clear, use the MaxL statement without the **physical** 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. For more information, refer to *Clear Data from Aggregate Storage Cubes*.

To remove cells with a value of zero, use the **alter database** MaxL statement with the **merge** grammar and the **remove_zero_cells** keyword.

Examples:

```
alter database ASOSamp.Basic clear data in region '{Jan, Budget}' physical;
```

Clears all Budget data for the month of Jan, using the physical method, from the ASOsamp.Basic database.

```
alter database ASOsamp.Basic clear data in region '{Jan, Budget}';
```

Clears all Budget data for the month of Jan, using the logical method, from the ASOsamp.Basic database.

```
alter database ASOsamp.Basic clear data in region '{CrossJoin({[Promotions].[Coupon]},{[Time].[1st Half]}), CrossJoin({[Promotions].[Coupon]},{[Time].[2nd Half]})}' physical;
```

Clears two physical regions from the ASOsamp.Basic database.

```
alter database ASOsamp.Basic clear data in region 'CrossJoin({Jan}, {Forecast1, Forecast2})';
```

Clears all January data for the Forecast1 and Forecast2 scenarios from the ASOsamp.Basic database.

alter database DBS-NAME compact outline

Compact the outline file to decrease the outline file size. Compaction helps keeps the outline file at an optimal size. After the outline file is compacted, the file continues to grow as before, when members are added or deleted.

Note:

Compacting the outline does not cause Essbase to clear the data. When a member is deleted from the outline, the corresponding record of that member in the outline file is marked as deleted, but the record remains in the outline file. Compacting the outline file does not remove the records of deleted members.

Example:

```
alter database ASOsamp.Basic compact outline;
```

alter database DBS-NAME add variable ...

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 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.

Example:

```
alter database ASOsamp.Basic add variable Month 'Oct';
```

alter database DBS-NAME drop variable ...

Remove a substitution variable and its corresponding value from the database.

Example:

```
alter database ASOsamp.Basic drop variable Month;
```

alter database DBS-NAME initialize load_buffer ...

Create a temporary buffer in memory for loading data.

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.

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.

- Specify **resource_usage** [RNUM](#) to put constraints on the load buffer.
- Specify **property** [PROPS](#) to tell Essbase what to do with empty values in the source data.
- Specify **wait_for_resources** to wait on processing load buffer operations until resources become available. The wait time is either the amount of time specified by `ASOLOADBUFFERWAIT` configuration, or the default wait, which is 10 seconds.

Example:

```
alter database ASOsamp.Basic 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.Basic database. The buffer can use only 50% of available resources. Missing values and zeros in the data source are ignored.

Refer also to [Load Data Using Buffers](#), Load Data into Aggregate Storage Cubes.

alter database DBS-NAME destroy load_buffer ...

Destroy the temporary data-load memory buffer.

Example:

```
alter database ASOsamp.Basic destroy load_buffer with buffer_id 1;
```

alter database DBS-NAME unlock all objects

Unlock all objects on the database that are in use by a user or process. You must be a service administrator. Service administrators can unlock any object, but other users can unlock only those objects that they locked.

Example:

```
alter database ASOSamp.Basic unlock all objects;
```

alter database DBS-NAME rename ...

Rename the database. When you rename a database, the cube directory is also renamed.

Example:

```
alter database ASOSamp.Basic rename to Basic2;
```

alter database DBS-NAME 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 Smart View or other grid client users, use `set note` instead.

Example:

```
alter database ASOSamp.Basic comment 'my comment';
```

alter database DBS-NAME merge all[incremental data [remove_zero_cells]

Merge incremental data slices. Use any of these keywords:

- **all**—Merge all incremental data slices into the main database slice.
- **incremental**—Merge all incremental data slices into a single data slice. The main database slice is not changed.
- (Optional) **remove_zero_cells**—When merging incremental data slices, remove cells that have a value of zero (logically clearing data from a region results in cells with a value of zero).

Examples:

```
alter database ASOsamp.Basic merge all data;
```

Merges all incremental data slices into the main slice in the ASOsamp.Basic database.

```
alter database ASOsamp.Basic merge incremental data;
```

Merges all incremental data slices into a single data slice within the ASOsamp.Basic database.

```
alter database ASOsamp.Basic merge all data remove_zero_cells;
```

Merges all incremental data slices into the main slice in the ASOsamp.Basic database, and removes cells with a value of zero.

alter database DBS-NAME 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.

Example:

```
alter database ASOSamp.Basic enable replication_assume_identical_outline;
```

To enable this functionality at the server or application level, use `REPLICATIONASSUMEIDENTICALOUTLINE` configuration instead.

alter database DBS-NAME 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.

Example:

```
alter database ASOSamp.Basic disable replication_assume_identical_outline;
```

alter database DBS-NAME begin archive to file

Prepare the database for backup by an archiving program, and prevent writing to the files during backup. You must be a system administrator to perform this action.

Begin archive achieves the following outcomes:

- 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 `end archive`.
- 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.

Begin archive and end archive do not perform the backup; they simply protect the database during the backup process.

 **Note:**

Using the **begin archive to file** and **end archive** grammar is the only supported way to backup and recover a database using MaxL.

Example:

```
alter database ASOSamp.Basic begin archive to file  
'asosamp_basic.arc';
```

alter database DBS-NAME end archive

Return the database to read-write mode after backing up the database files.

 **Note:**

Using the **begin archive to file** and **end archive** grammar is the only supported way to backup and recover a database using MaxL.

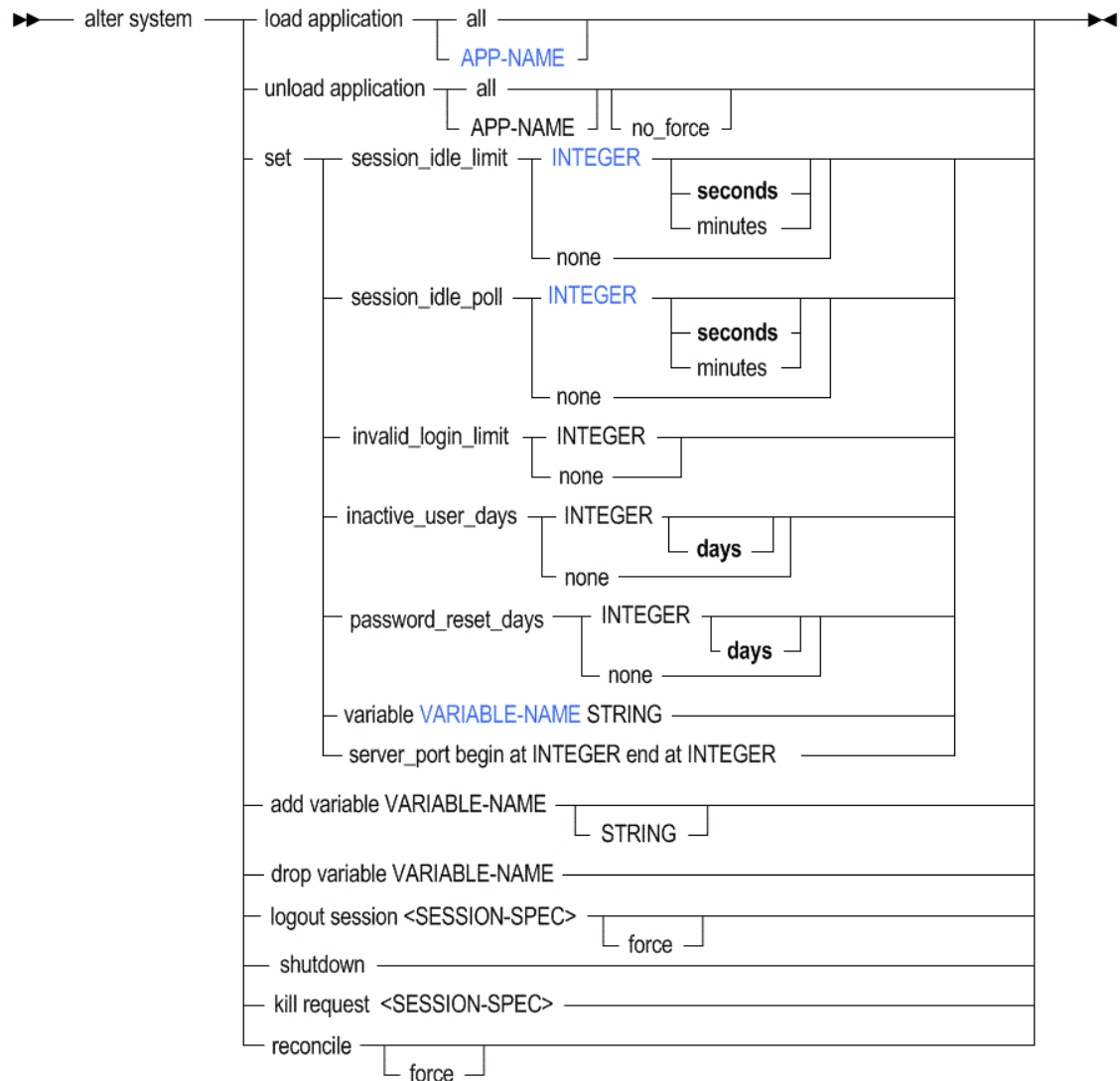
Alter System (Aggregate Storage)

The MaxL alter system statement helps you change the state of the Essbase Server.

[Click here for non-aggregate storage version](#)

Use this statement to start and stop applications, manipulate Essbase Server-wide variables, manage password and login activity, disconnect users, kill processes, and shut down the server.

Syntax



- APP-NAME
- INTEGER
- VARIABLE-NAME

Keywords

Use MaxL **alter system** to change the following Essbase Server-wide settings. The role required for most of the statements is System Administrator, with exceptions noted.

alter system load application ...

Start an application, or start all applications on the Essbase Server. Requires at least Database Access permission for the application.

Example:

```
alter system load application ASOSamp;
```

alter system unload application ...

Stop an application, or stop all applications on the Essbase Server. Requires at least Database Access permission for the application. Unloading an application cancels all active requests and database connections, and stops the application. If Essbase encounters a problem when trying to cancel active requests and database connections, and stopping the application, an error is logged in the application log.

If you do not want to stop an application if it has active requests and database connections, use the **no_force** keyword. When using **no_force**:

- If the application has active requests and database connections, the application is not stopped; it continues running.
- If the application does not have active requests and database connections, the application is stopped, as if you used **unload application** without specifying **no_force**.

Note:

Unloading an application cancels all active requests and database connections, and stops the application, unless you specify otherwise using **no_force**. **no_force** causes Essbase to return an error if active requests are running on the application. An internal logic error [200] is logged when a database is unable to shut down gracefully when unloading an application or shutting down the system while a process is running on the database.

Example:

```
alter system unload application ASOSamp no_force;
```

alter system set session_idle_limit ...

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.

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.

Because users may mistakenly assume that the calculation stopped when they were logged out, you can do one of the following to correct the user experience:

- 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

Example:

```
alter system set session_idle_limit 60 minutes;
```


alter system set session_idle_poll ...

Set the time interval for inactivity checking. The time interval specified in the session idle poll tells Essbase how often to check whether user sessions have passed the allowed inactivity interval indicated by `session_idle_limit`.

Example:

```
alter system set session_idle_poll 300 seconds;
```

alter system set invalid_login_limit ...

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.

Example:

```
alter system set invalid_login_limit 3;
```

alter system set inactive_user_days ...

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.

Example:

```
alter system set inactive_user_days 30;
```

alter system set password_reset_days ...

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 limit is set to **none**, there is no built-in limit for password retention. By default, there is no limit.

Example:

```
alter system set password_reset_days 60;
```

alter system set variable ...

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 (&).

Example:

```
alter system set variable Month Nov;
```

alter system set server_port ...

Expand the port range specified by Essbase configuration properties. Each Essbase application uses two ports from this range. If no more ports are available, an error message is displayed.

 **Note:**

You can expand port ranges only so that the beginning port range is less than SERVERPORTBEGIN and the ending port range is greater than SERVERPORTEND.

Example:

```
alter system set server_port begin at 32750 end at 33780;
```

alter system 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.

Example:

```
alter system add variable CurYear '2024';
```

alter system drop variable ...

Remove a substitution variable and its corresponding value from the system.

Example:

```
alter system drop variable CurYear;
```

alter system logout session all

Terminate all user sessions currently running on the Essbase Server.

Example:

```
alter system logout session all;
```

alter system 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.

Example:

```
alter system logout session 3694133190 force;
```

alter system logout session SESSION-ID

Terminate a session by its unique session ID number. To see the session ID number, use [display session](#).

Example:

```
alter system logout session 3694133190;
```

alter system logout session by user USER-NAME

Terminate all current sessions by a particular user, either across the entire Essbase Server, or limited to a specific application or database.

Example:

```
alter system logout session by user User1;
```

alter system logout session by user ... on application ...

Terminate all current sessions by a particular user across a specific application.

Example:

```
alter system logout session by user User1 on application ASOSamp;
```

alter system logout session by user on database

Terminate all current sessions by a particular user across a specific database.

Example:

```
alter system logout session by user User1 on database ASOSamp.Basic;
```

alter system logout session on application

Terminate all current user sessions across a specific application.

Example:

```
alter system logout session on application ASOSamp;
```

alter system logout session on database

Terminate all current user sessions across a specific database.

Example:

```
alter system logout session on database ASOSamp.Basic;
```

alter system shutdown

Shut down the Essbase Server.

Example:

```
alter system shutdown;
```

alter system kill request all

Terminate all current requests on the Essbase Server.

**Note:**

To terminate your own active request in MaxL Shell, press the ESC key.

Example:

```
alter system kill request all;
```

alter system kill request SESSION-ID

Terminate the current request indicated by the session ID. You can obtain session IDs using [display session](#).

Example:

```
alter system kill request 3694133190;
```

alter system kill request by user

Terminate all current requests by the specified user on the Essbase Server.

Example:

```
alter system kill request by user User1;
```

alter system kill request on application

Terminate all current requests on the specified application.

Example:

```
alter system kill request on application ASOSamp;
```

alter system kill request on database

Terminate all current requests on the specified database.

Example:

```
alter system kill request on database ASOSamp.Basic;
```

alter system reconcile

No longer applicable.

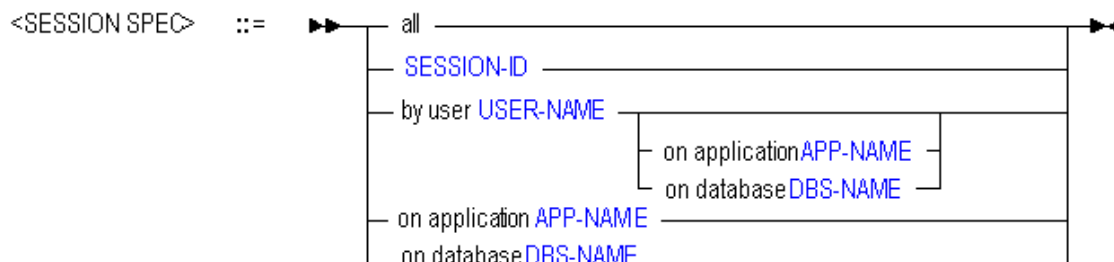
Notes

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.



- [SESSION-ID](#)

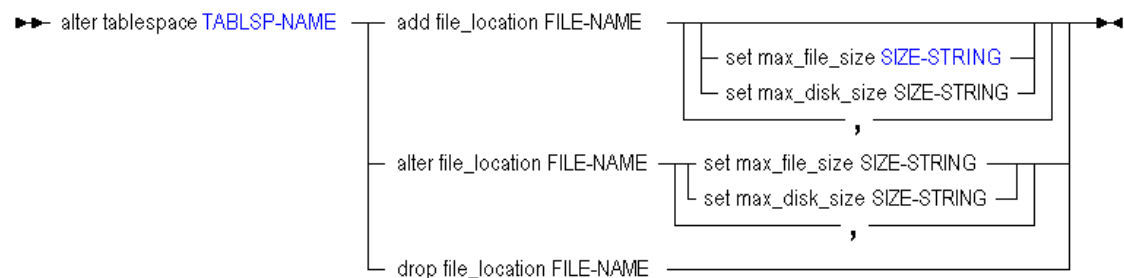
- `USER-NAME`
- `APP-NAME`
- `DBS-NAME`
- `APP-NAME`
- `DBS-NAME`

Alter Tablespace (Aggregate Storage)

The MaxL **alter tablespace** statement helps you update tablespace details for an Essbase aggregate storage (ASO) database. Tablespaces are not applicable to block storage cubes. The minimum permission required to edit ASO tablespace information is Application Manager.

To get a list of tablespaces, use [display tablespace](#). You cannot change the location or size of the metadata and log tablespaces.

Syntax



- `TABLSP-NAME`
- `SIZE-STRING`

Keywords

You can use MaxL **alter tablespace** to edit ASO tablespaces in the following ways.

alter tablespace TABLSP-NAME add file_location ...

Add a new file location to the tablespace.

Example:

```
alter tablespace ASOsamp.'default' add file_location 'C:\\mytablespace';
```

Adds another file location for the default tablespace.

alter tablespace TABLSP-NAME alter file_location ...

Change the maximum file-size or disk-size value for the specified file location.

Note:

FILE-NAME is case sensitive in this statement.

alter tablespace TABLSP-NAME alter file_location ... set max_file_size ...

Specify a value for the maximum size that a data file may attain before Essbase creates a new file.

The largest possible value that the ASO 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.

The minimum value is 8MB (8388608b), and any values you enter are rounded up to the next 8MB interval.

**Note:**

Some operating system platforms may enforce a maximum file size limit.

Examples:

```
alter tablespace ASOsamp.'default' alter file_location 'C:\\Oracle\\
\\Middleware\\OracleHome\\essbase\\products\\Essbase\\EssbaseServer\\' set
max_file_size 128mb;
```

Changes the maximum file size allowed in the specified location of the default tablespace. Note that the file_location string is case sensitive.

```
alter tablespace ASOsamp.'default' alter file_location '\\\\ComputerName\\
\\SharedFolder\\Resource' set max_file_size 128mb;
```

Changes the maximum file size allowed in the specified location of the default tablespace. The file_location string is specified using UNC.

alter tablespace TABLSP-NAME alter file_location ...set max_disk_size ...

Specify the value for the maximum amount of disk space to be allocated to the file location. The largest possible value that the ASO 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. The minimum value is 8MB (8388608b), and any values you enter are rounded up to the next 8MB interval.

Example:

```
alter tablespace ASOsamp.'default' alter file_location 'C:\\Oracle\\
\\Middleware\\OracleHome\\essbase\\products\\Essbase\\EssbaseServer\\' set
max_disk_size 128mb;
```

alter tablespace TABLSP-NAME drop file_location ...

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.



Note:

FILE-NAME is case sensitive in this statement.

Example:

```
alter tablespace ASOsamp.'default' drop file_location 'C:\\mytablespace';
```

Notes

- This statement requires the application to be started.
- On Windows, you can specify tablespace file locations using Uniform Naming Convention (UNC) syntax, which is \\ComputerName\SharedFolder\Resource. Including the escape characters required by MaxL Shell, the UNC file name specification would look like the following:

```
'\\\\\\ComputerName\\SharedFolder\\Resource'
```

Create Application (Aggregate Storage)

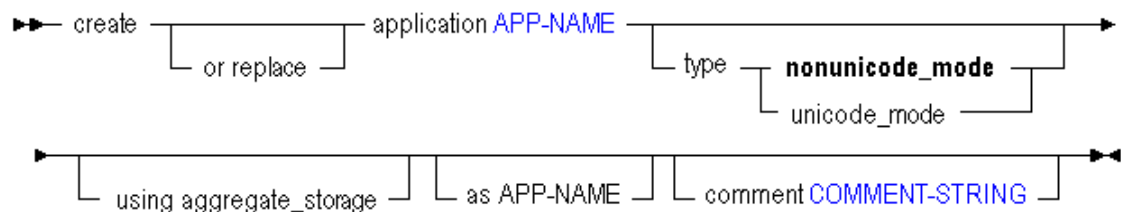
The MaxL **create application** statement for aggregate storage (ASO) helps you create or re-create an application. Every Essbase database (cube) must be created within an application. If the cube is intended to be aggregate storage, then the application must be specified as aggregate storage.

[Click here for non-aggregate storage version](#)

You can create the application either from scratch or as a copy of another application on the same server. The permission required to create an application is power user or system administrator.

See [APP-NAME](#) for information on the maximum length of and special characters that are allowed in an application name. Application names are not case-sensitive.

Syntax



- [APP-NAME](#)
- [COMMENT-STRING](#)

Keywords

Use the MaxL **create application** statement to create an aggregate storage (ASO) application in the following ways.

create application APP-NAME using aggregate_storage

Create a new application intended for a database (cube) that uses aggregate storage (ASO) mode. Only one database per application is allowed. Selecting to use aggregate storage for the application is non-reversible.

Choose an ASO application if the following requirements for your database are true:

- 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 procedural calculation scripts.

Example:

```
create application MyASOApp using aggregate_storage;
```

create or replace application ...

Create an application intended for an ASO database, replacing an existing application of the same name, if one exists.

Example:

```
create or replace application MyASOApp using aggregate_storage;
```

create application APP-NAME type nonunicode_mode ...

Create a non-Unicode-mode application intended for an ASO database. Non-Unicode mode is the default application type that is created, even if these keywords are omitted.

Example:

```
create application MyASOApp type nonunicode_mode using aggregate_storage;
```

create application APP-NAME type unicode_mode ...

Create a Unicode-mode application intended for an ASO database.

Example:

```
create application MyASOApp type unicode_mode using aggregate_storage;
```

create application APP-NAME ... as

Create an ASO application that is a copy of another ASO application. To copy an application, Application Manager permission on the source application is required, in addition to the power user role required to create an application.

You cannot copy block storage applications to aggregate storage applications or vice versa.

The copy will always use the same storage mode as the original. However, you can convert an outline from a block storage database to an aggregate storage database, using [create outline](#).

Before you copy an aggregate storage application, you must merge all incremental data slices into the main database slice. Data in unmerged incremental data slices is not copied.

Example:

```
create application MyASOApp using aggregate_storage as ASOSamp;
```


Creates an aggregate storage application called MyASOApp, which is a copy of the gallery sample application named ASOSamp.

create application APP-NAME ... comment

Create an ASO application including a short description (optional). The description can contain up to 80 characters.

Example:

```
create application MyASOApp using aggregate_storage as ASOSamp comment 'ASO
app based on ASOSamp';
```

Create Database (Aggregate Storage)

The MaxL create database statement for ASO mode helps you create or re-create an aggregate storage Essbase cube, optionally as a copy of another on the same server.

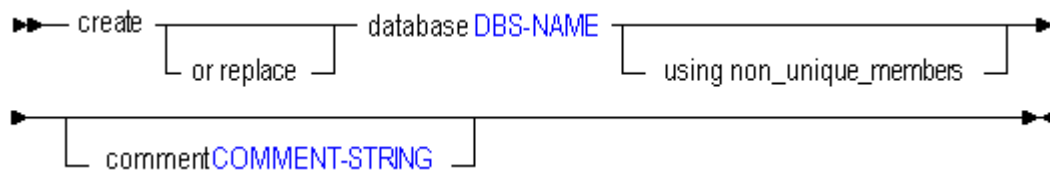
[Click here for non-aggregate storage version](#)

See [DBS-NAME](#) for information on the maximum length of and special characters that are allowed in a database name. Database names are not case-sensitive.

The syntax for creating an aggregate storage database is the same as for creating a block storage (BSO) database, but you must create the aggregate storage database within an existing aggregate storage application.

The minimum permission required to create a database is Application Manager. To copy a database, at least Database Manager permission on the source database is also required.

Syntax



- [DBS-NAME](#)
- [COMMENT-STRING](#)

Keywords

Use the MaxL **create database** statement to create an Essbase cube in the following ways:

create database DBS-NAME

Create a new database.

Example:

```
create database MyASOApp.Basic;
```

create or replace database DBS-NAME

Create a database, or replace an existing database of the same name.

Example:

```
create or replace database MyASOApp.Basic;
```

Creates a database called Basic within the MyASOApp application. If a database named Basic within the MyASOApp application already exists, it is overwritten.

create database DBS-NAME 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.

Example:

```
create database MyASOApp.Basic using non_unique_members;
```

For more information about duplicate member names, see [Creating and Working With Duplicate Member Outlines](#).

create database DBS-NAME as ...

Create a database as a copy of another database.

Example:

```
create database MyASOApp.New as ASOSamp.Basic;
```

Creates a database called New within the MyASOApp application that is a copy of the database Basic within the ASOSamp application.

create database DBS-NAME ... comment

Create a database description (optional). The description can contain up to 80 characters.

Example:

```
create database MyASOApp.Basic comment 'New ASO cube';
```

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 [Create an Aggregate Storage Sample Using MaxL](#).

Create Outline (Aggregate Storage)

The MaxL **create outline** statement for aggregate storage cubes helps you create an Essbase aggregate storage (ASO) outline that is based on an existing block storage (BSO) outline.

The outline you are creating must be for an ASO cube that is local to your current login session. The BSO cube 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.

The minimum permission required to create an ASO outline is Database Manager.

Essbase supports the following outline conversion scenarios:

- Convert a non-Unicode BSO outline to a non-Unicode ASO outline.
- Convert a non-Unicode BSO outline to a Unicode ASO outline.
- Convert a Unicode BSO outline to a Unicode ASO outline.

The following outline conversion scenarios are not supported:

- Converting a Unicode BSO outline to a non-Unicode ASO outline.
- Converting an ASO outline to a BSO 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 ]
  
```

- DBS-NAME
- HOST-NAME
- USER-NAME
- PASSWORD

Keywords

You can convert an outline in the following ways using MaxL **create outline**.

create outline ...

Create an aggregate-storage cube outline based on a block storage outline. If an outline of the same name already exists, it is replaced.

Example:

```
create outline on aggregate_storage database MyASOApp.Basic2 as outline on
database Sample.Basic;
```

create or replace outline ...

This statement has the same result as `create outline` above.

Example:

```
create or replace outline on aggregate_storage database MyASOApp.Basic2 as
outline on database Sample.Basic;
```

create outline ... at HOST-NAME

If the block-storage cube you are using as a source is remote, specify its discovery URL (ending in `/agent`). For example, `"https://myserver.example.com:9001/essbase/agent"`.

Example:

```
create outline on aggregate_storage database MyASOApp.Basic2 as outline on
database Sample.Basic at "https://myserver.example.com:9001/essbase/agent";
```

create outline ... as USER-NAME identified by PASSWORD

If the block-storage cube you are using as a source is remote, specify the location. 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 outline on aggregate_storage database MyASOApp.Basic2 as outline on
database Sample.Basic at "https://myserver.example.com:9001/essbase/agent" as
'AppMgrUsr' identified by 'pa55w0rD';
```

See Also

For a complete example of how to create an ASO version of a block storage cube, refer to [Creating an Aggregate Storage Sample Using MaxL](#).

Display Tablespace (Aggregate Storage)

The MaxL display tablespace statement for ASO mode helps you view details about a tablespace belonging to an Essbase aggregate storage database.

Tablespaces are applicable only to aggregate storage (ASO) cubes.

The minimum permission required to run this statement is Application Manager.

This statement requires the application to be started.

Syntax

```
►► display tablespace TABLSP-NAME ◄◄
```

TABLSP-NAME

Example

```
set column_width 50; /* so file_location will not be truncated */
display tablespace ASOSamp.'default';
```

Table 3-9 Display Tablespace MaxL Output Columns

Column Header	Contents
file_location	/scratch/oraclehome/applications/essbase/app
max_file_size	134217727
max_disk_size	4294967295

Execute Aggregate Build (Aggregate Storage)

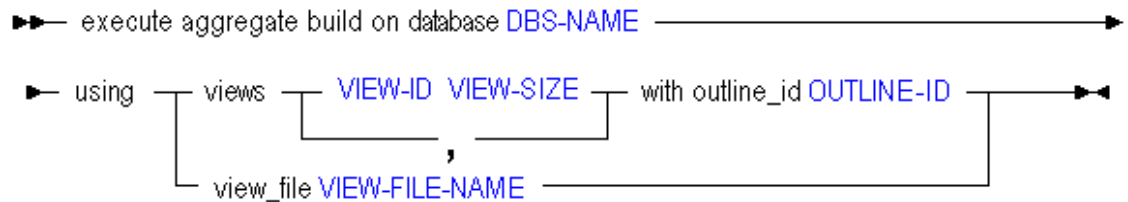
The MaxL **execute aggregate build** statement helps you perform an aggregation on an Essbase aggregate storage (ASO) cube, based on selected views.

The minimum application permission required to materialize ASO aggregations is Application Manager.

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.

You can also configure Essbase to generate aggregate views automatically. For more information about aggregate views, refer to [Aggregating an Aggregate Storage Database](#).

Syntax



- DBS-NAME
- VIEW-ID
- VIEW-SIZE
- OUTLINE-ID
- VIEW-FILE-NAME

Keywords

You can build (materialize) ASO aggregations in the following ways using MaxL **execute aggregate build**.

execute aggregate build ... using views...

Builds an aggregation based on a previously selected view (or views) and the associated ASO outline ID.

execute aggregate build ... using view_file VIEW-FILE-NAME

Builds an aggregation based on a saved view selection stored in an aggregation script. Omit the `.csc` file extension from the view file name when you issue the **execute aggregate build** 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.

Examples

```
execute aggregate build on database ASOSamp.Basic using views 711 0.00375
with outline_ID 4142187876;
```

Builds an aggregation of the ASOSamp.Basic cube. 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 ASOSamp.Basic using view_file myView;
```

Builds an aggregation of the ASOSamp.Basic database based on the view saved in the aggregation script `myView.csc`.

Related Topics

- [Execute Aggregate Process \(Aggregate Storage\)](#)
The MaxL **execute aggregate process** statement helps you perform an aggregation on an Essbase aggregate storage (ASO) cube.
- [Execute Aggregate Selection \(Aggregate Storage\)](#)
The MaxL **execute aggregate selection** statement helps you select views of an Essbase aggregate storage cube, based on various criteria.

Execute Aggregate Process (Aggregate Storage)

The MaxL **execute aggregate process** statement helps you perform an aggregation on an Essbase aggregate storage (ASO) cube.

The minimum application permission required to perform an ASO aggregation is Application Manager.

Use this statement to perform an aggregation. Optionally, you can specify the maximum disk space for the resulting files. Optionally, you can base the view selection on user querying patterns.

This statement applies to aggregate storage cubes only.

The **execute aggregate process** statement enables you to build ASO aggregate views with a minimum of interactive settings. If greater control is needed, you can instead combine the following statements to select and materialize the aggregations:

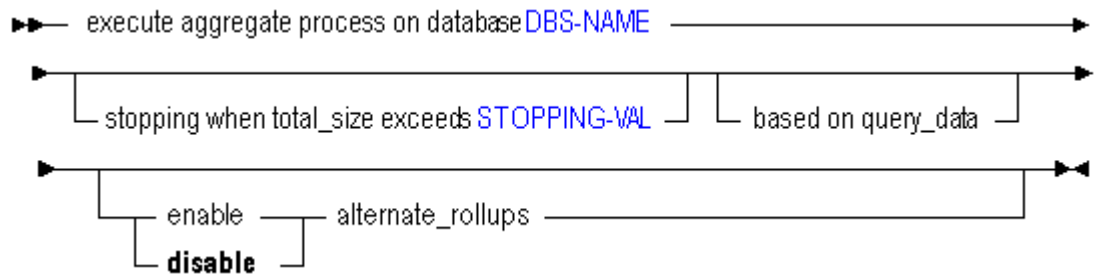
- [Execute Aggregate Selection \(Aggregate Storage\)](#)
- [Execute Aggregate Build \(Aggregate Storage\)](#)

When you run **execute aggregate process** on an ASO cube, it 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.

You can also configure Essbase to generate aggregate views automatically. For more information about aggregate views, refer to [Aggregating an Aggregate Storage Database](#).

Syntax



- [DBS-NAME](#)
- [STOPPING-VAL](#)

Keywords

You can explicitly initiate an aggregation of an ASO cube in the following ways using MaxL **execute aggregate process**.

execute aggregate process ... stopping when total_size exceeds...

Aggregate whichever views Essbase selects, with the exception that the maximum growth of the aggregated cube must not exceed the given ratio. For example, if the size of a cube is 1 GB, specifying the total size as 1.2 means that the size of the resulting data cannot exceed 20% of 1 GB, for a total size of 1.2 GB.

execute aggregate process ... 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 [alter database](#) with the **enable query_tracking** grammar.

execute aggregate process ... enable|disable alternate_rollups

If enabled, secondary hierarchies (with default level usage) are considered for view selection. By default, 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](#).

Examples

```
execute aggregate process on database ASOsamp.Basic
stopping when total_size exceeds 1.3;
```

Selects and builds an aggregation of the ASOsamp.Basic cube that permits the cube to grow by no more than 30% as a result of the aggregation.

```
execute aggregate process on database ASOsamp.Basic based on query_data;
```

Selects and builds an aggregation of the ASOsamp.Basic cube, where the views that Essbase selects for aggregation are based on the most frequently queried areas of the cube.

Related Topics

- [Execute Aggregate Build \(Aggregate Storage\)](#)
The MaxL **execute aggregate build** statement helps you perform an aggregation on an Essbase aggregate storage (ASO) cube, based on selected views.
- [Execute Aggregate Selection \(Aggregate Storage\)](#)
The MaxL **execute aggregate selection** statement helps you select views of an Essbase aggregate storage cube, based on various criteria.

Execute Aggregate Selection (Aggregate Storage)

The MaxL **execute aggregate selection** statement helps you select views of an Essbase aggregate storage cube, based on various criteria.

The minimum application permission required to select views for ASO aggregation is Application Manager.

After you use this statement to select views, you use the resultant table or aggregation script to build an aggregation (materialize a view) using [execute aggregate build](#).

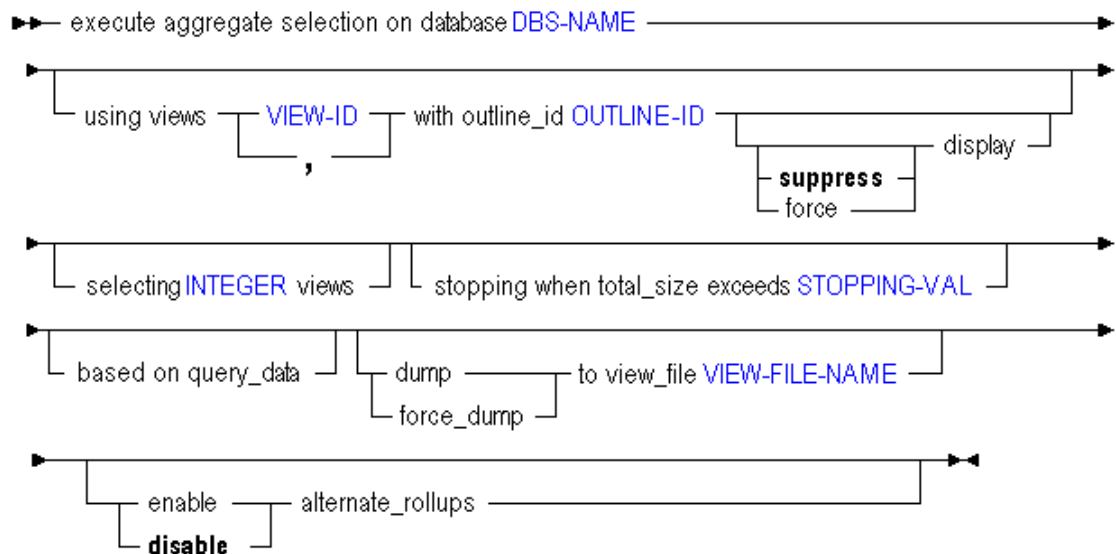


Note:

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.

You can also configure Essbase to generate aggregate views automatically. For more information about aggregate views, see [Aggregating an Aggregate Storage Database](#).

Syntax



- [DBS-NAME](#)

- **OUTLINE-ID**
- **VIEW-ID**
- **INTEGER**
- **STOPPING-VAL**
- **VIEW-FILE-NAME**

Keywords

You can select ASO aggregate views (for subsequent aggregation) in the following ways using MaxL **execute aggregate selection**.

execute aggregate selection ... using views ... with outline_ID ...

Selects views based on pre-selected view IDs. The view IDs are obtained from previous executions of the statement.

execute aggregate selection ... using views ... with outline_ID ... force display

Selects views based on pre-selected view IDs, including the pre-selected views IDs themselves.

execute aggregate selection ... 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.

execute aggregate selection ... selecting INTEGER views

Selects the number of views based on whether the number of views specified in **INTEGER** is greater than or equal to, or less than, the recommended number of default views that are returned by the **execute aggregate selection** statement. By default, Essbase determines the recommended number of default views.

Assume that **RECNUM** represents the recommended number of default views:

- If the value of **INTEGER** is greater than or equal to the value of **RECNUM**, the selected number of views equals **RECNUM**.

For example, if **INTEGER** equals 20 and **RECNUM** equals 15, the number of selected number of views equals 15.

- If the value of **INTEGER** is less than the value of **RECNUM**, the number of views that are selected equals **INTEGER**.

If you want the number of views that are selected to equal the value of **INTEGER**, use the **stopping when total_size exceeds STOPPING-VAL** clause to change the number of recommended default views that are returned by the **execute aggregate selection** statement. Define the **STOPPING-VAL** factor large enough so that the number of default views that are returned by **execute aggregate selection** is greater than the value of **INTEGER**.

For example, if **INTEGER** equals 20 and **RECNUM** equals 50, the number of selected number of views equals 20.

Note:

This parameter does not create views.

execute aggregate selection ... stopping when total_size exceeds STOPPING-VAL

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 cube to grow by no more than 50% as a result of the aggregation.

execute aggregate selection ... based on query_data

Selects views based on previously collected query-tracking data. You must have already enabled query tracking. After enabling query tracking, allow sufficient time to collect user data-retrieval patterns before performing an aggregate selection based on query data.

Query tracking records information about every query executed on the cube, 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.

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 **alter database DBS-NAME disable query_tracking**. In both cases, all the query cost data is discarded, and the recording stops (and will not continue when the application starts again). All query cost data becomes invalid when additional views are built.

To create views based on tracked query patterns,

1. If needed, enable query tracking using **alter database DBS-NAME enable query_tracking**. Query tracking is on by default.
2. Run all production queries once, and then select the first set of views based on the query cost data. To select the views, run this MaxL statement (**execute aggregate selection... based on query_data...**).
3. Build the selected aggregate view using **execute aggregate build**.
4. Repeat the previous two steps at least twice. Selecting and building multiple views iteratively helps ensure there are enough usage-tracking data to form a pattern. Each new view you build decreases the rate at which query costs grow.

execute aggregate selection ... dump to view_file

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 **force_dump** keyword.

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: `execute aggregate build on database <db-name> using view_file <view-file-name>`

execute aggregate selection ... force_dump to view_file

Saves the view selection to an aggregation script. If the specified script name already exists, the **force_dump** keyword causes it to be overwritten.

execute aggregate selection ... enable|disable alternate_rollups

If enabled, secondary hierarchies (with default level usage) are considered for view selection. Default: disabled (no secondary hierarchies are considered).

Examples

```
execute aggregate selection on database ASOsamp.Basic;
```

Performs the default view selection for ASOsamp Basic. This statement selects the same views as `execute aggregate process` on database `ASOsamp.Basic` would build.

```
execute aggregate selection on database ASOsamp.Basic 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.Basic 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.Basic stopping when  
total_size exceeds 1.2;
```

Selects an aggregation of the ASOsamp Basic cube that, when built, would permit the cube to grow by no more than 20% as a result of the aggregation.

```
execute aggregate selection on database ASOsamp.Basic based on query_data;
```

Selects views based on previously collected query-tracking data. You must have enabled query tracking using **alter database DBS-NAME enable query_tracking**.

```
execute aggregate selection on database ASOsamp.Basic  
dump to view_file myView;
```

Selects a default aggregation of the ASOsamp Basic cube, 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.Basic using view_file  
'myView.csc';
```

Related Topics

- [Execute Aggregate Build \(Aggregate Storage\)](#)
The MaxL **execute aggregate build** statement helps you perform an aggregation on an Essbase aggregate storage (ASO) cube, based on selected views.

- [Execute Aggregate Process \(Aggregate Storage\)](#)
The MaxL **execute aggregate process** statement helps you perform an aggregation on an Essbase aggregate storage (ASO) cube.
- [Alter Database \(Aggregate Storage\)](#)
The MaxL alter database statement for ASO mode helps you change Essbase database-wide settings.

Execute Allocation (Aggregate Storage)

The MaxL **execute allocation** statement helps you allocate one or more given source amounts to a target range of cells in an Essbase aggregate storage (ASO) database.

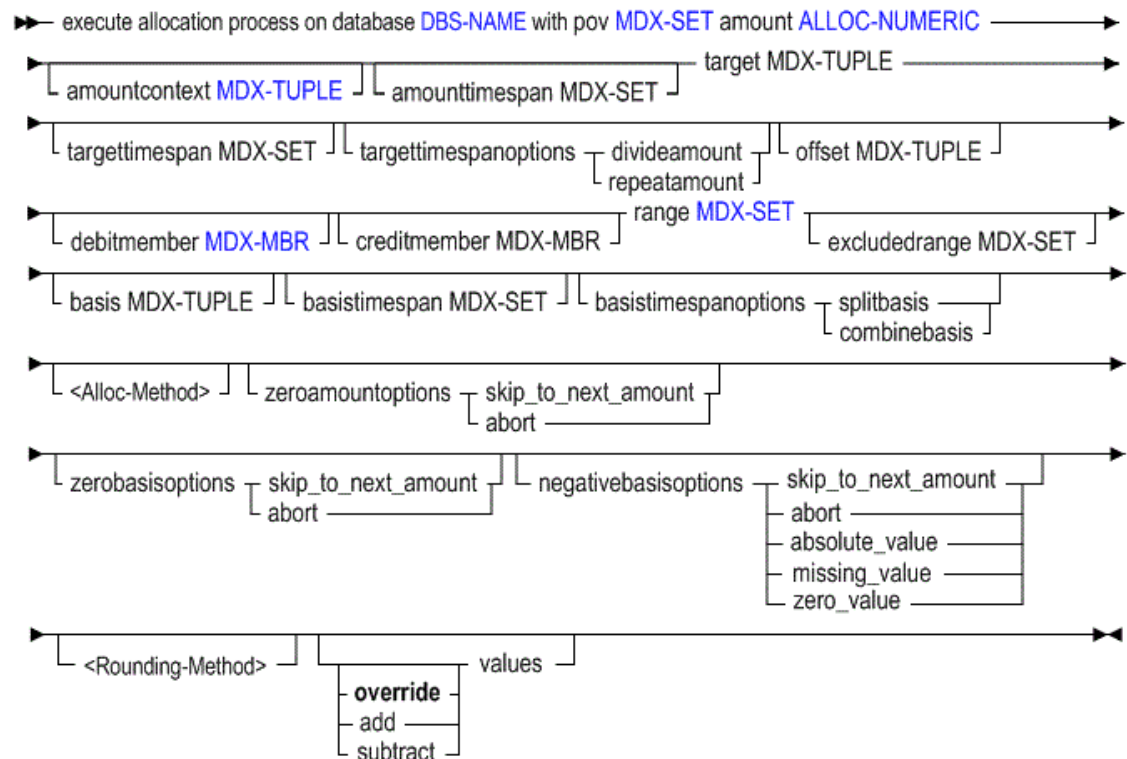
The source amount can be allocated to the target proportionately to a given basis, or the source amount can be spread evenly to the target region.

Allocations are typically used in the budgeting process to distribute revenues or costs.

The minimum application permission required to run an ASO custom calculation is Database Update.

For more information about allocations and to understand the input parameters, see [Performing Custom Calculations and Allocations on Aggregate Storage Databases](#).

Syntax



- **DBS-NAME**
- **MDX-SET**
- **ALLOC-NUMERIC**
- **MDX-TUPLE**

- MDX-MBR

Keywords

You can run ASO custom allocations in the following ways using MaxL **execute allocation**.

execute allocation ... pov MDX-SET

Required. Provide an MDX set defining the context region in which the allocation should be performed.

execute allocation ... amount ALLOC-NUMERIC

Required. Provide an MDX numeric value expression indicating the amount to be allocated.

execute allocation ... 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.

execute allocation ... 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.

execute allocation ... target MDX-TUPLE

Required. Provide an MDX tuple defining the database region where the allocation results should be written.

execute allocation ... targettimespan MDX-SET

Optional. Provide an MDX set indicating one or more time periods to be considered for the allocation target. Time periods must be level 0 members in a Time dimension.

execute allocation ... targettimespanoptions

Optional, but required if `targettimespan` is used.

Select a method for allocating values across the target time span:

- `divideamount`—Divide the amount evenly across the time periods.
- `repeatamount`—Repeat the amount across the time periods.

execute allocation ... offset MDX-TUPLE

Optional. If you use offsetting entries (general ledger counterbalancing measures), provide an MDX tuple defining the location in the database where an offsetting value should be written for each source amount.

execute allocation ... debitmember MDX-MBR

Optional. If you use double-entry accounting, provide an MDX member expression indicating the member to which positive result values should be written.

execute allocation ... creditmember MDX-MBR

Optional. If you use double-entry accounting, provide an MDX member expression indicating the member to which negative result values should be written.

execute allocation ... range MDX-SET

Required. Provide an MDX set indicating the database region in which allocated values are calculated and written.

execute allocation ... excludedrange MDX-SET

Optional. Provide an MDX set specifying locations in the range where you do not want allocation values written.

execute allocation ... 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.

execute allocation ... 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.

execute allocation ... basistimespanoptions

Optional, but required if **basistimespan** is used. Select a method for using the basis time span:

- `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**.

execute allocation ... share

Optional. For the allocation method, allocate the amount(s) proportionately to the basis values. For syntax, see Allocation Method Specification in Notes.

execute allocation ... spread

Optional. For the allocation method, 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:

- `skip_missing`—Skip missing basis values.
- `skip_zero`—Skip zero basis values.
- `skip_negative`—Skip negative basis values.

execute allocation ... zeroamountoptions

Optional. If this syntax is omitted, zero or #MISSING amount values are allocated. If you don't want to allocate empty values, specify how to handle them:

- `skip_to_next_amount`—Skip to the next nonzero, non-#MISSING amount value.
- `abort`—Cancel the entire allocation operation.

execute allocation ... zerobasisoptions

Optional. For **share** allocation method, this option specifies the action when the sum of all basis values is zero. For **spread** allocation method, this option specifies the action when all the basis values are skipped. Select one of the following options:

- `skip_to_next_amount`—Skip to the next nonzero, non-#MISSING amount value.
- `abort`—Cancel the entire allocation operation.

execute allocation ... round

Optional rounding specification. Choose INTEGER or MDX-NUMERIC, plus one of the subsequent options, to specify the rounding method.

- `round INTEGER|MDX-NUMERIC`—Round to a specified number of decimal places, using an integer or an MDX numeric value expression. The value must be between 100 and -100, and is truncated if it is not a whole number.
- `discard errors`—Perform rounding, but discard rounding errors.
- `errors_to_lowest`—Add rounding errors to the lowest allocated value.
- `errors_to_highest`—Add rounding errors to the highest allocated value.
- `errors_to_location MDX-TUPLE`—Provide an MDX tuple indicating a cell to which the rounding error should be added.

execute allocation ... [override][add][subtract values

Optional. Values generated by an ASO custom allocation can be added to (or subtracted from) existing values, instead of overwriting them. Overwriting is the default.

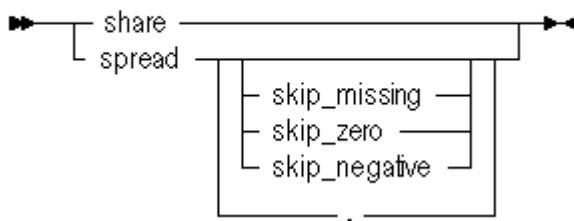
When performing custom allocations on an aggregate storage cube with a federated partition, you can only override existing values. You cannot add to, nor subtract from, existing values.

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.

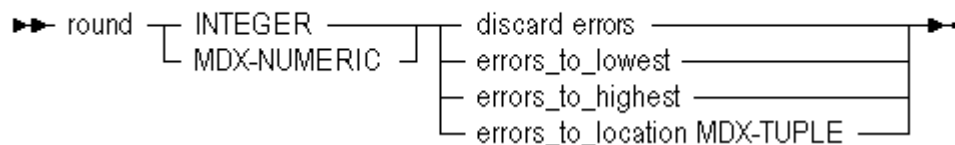
Allocation Method Specification

<Alloc-Method> ::=



Rounding Method Specification

<Rounding-Method> ::=



Example

The following statement executes an allocation.

```

execute allocation process 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])"
add values;

```

Execute Calculation (Aggregate Storage)

The MaxL **execute calculation** statement for ASO mode helps you run a custom calculation script for an Essbase aggregate storage database.

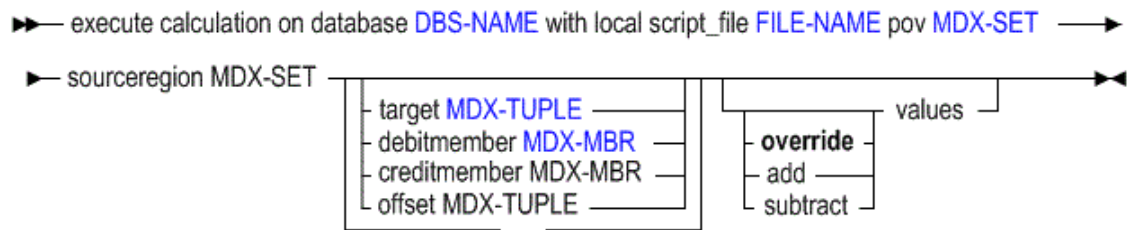
[Click here for non-aggregate storage version](#)

Use this statement to execute a custom calculation script expressed in MDX, specifying the calculation script file, source region, and point of view (POV). Optionally, specify the target, offset, and debit or credit members.

The minimum application permission required to run an ASO custom calculation is Database Update.

For more information about custom calculation script parameters, see [Performing Custom Calculations and Allocations on Aggregate Storage Databases](#).

Syntax



- **DBS-NAME**
- **FILE-NAME**
- **MDX-SET**
- **MDX-TUPLE**
- **MDX-MBR**

Keywords

You can run ASO custom calcs in the following ways using MaxL **execute calculation**.

execute calculation ... with 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, `script.txt`.

```
(AccountA,Proj1) := 100;
([AccountB], [Proj1]) := ([AccountB], [Proj1]) * 1.1;
(AccountC,Proj1) :=
    ((AccountB,Proj1,2007) + (AccountB, Proj1)) / 2;
(AccountA,Proj2) :=
    ((AccountD,Proj1) +
    (AccountB,Proj2)) / 2;
```

For information about writing custom calculation scripts, see *Performing Custom Calculations and Allocations on Aggregate Storage Databases*.

execute calculation ... pov MDX-SET

Required. Provide an MDX set defining the context region in which the calculation should be performed. The calculation script will be executed once for every cross-product in the POV region.

execute calculation ... 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.

execute calculation ... target MDX-TUPLE

Optional. Provide an MDX tuple defining the database region where results should be written. You can use only stored, level-0 members in the tuple; do not use upper-level members, attribute members, or dynamic calc members.

execute calculation ... debitmember MDX-MBR

Optional. If you use double-entry accounting, provide an MDX member expression indicating the member to which positive result values should be written. You can specify only stored,

level-0 members; do not use upper-level members, attribute members, or dynamic calc members.

execute calculation ... creditmember MDX-MBR

Optional. If you use double-entry accounting, provide an MDX member expression indicating the member to which negative result values should be written. You can specify only stored, level-0 members; do not use upper-level members, attribute members, or dynamic calc members.

execute calculation ... offset MDX-TUPLE

Optional. If you use offsetting entries (general ledger counterbalancing measures), provide an MDX tuple defining the location in the database where the offsetting value for each source amount should be written. You can use only stored, level-0 members in the tuple; do not use upper-level members, attribute members, or dynamic calc members.

execute calculation ... override|add|subtract values

Optional. Values generated by an ASO custom calculation can be added to (or subtracted from) existing values, instead of overwriting them. Overwriting is the default.

Notes

- Each clause can only be entered once.
- The **script_file**, **pov**, and **sourceregion** clauses are mandatory; the others are optional.
- The optional clauses following the **sourceregion** specification can be entered in any order, each separated by white space.
- 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](#).

```
execute calculation on database app.db with
  local script_file "script.txt"
  POV "Crossjoin({[VisionUS]},
    Crossjoin({[101]},
      Crossjoin ({[Jan]},
        Crossjoin({[Scenario]},
          Descendants(Geography, Geography.Levels(0))))))"
  SourceRegion "Crossjoin({[AccountB], [AccountD]},
    Crossjoin({[Proj1], [Proj2]}, {[2007]}))"
  Target "(Allocation)"
  DebitMember "[BeginningBalance_Debit]"
  CreditMember "[BeginningBalance_Credit]"
  Offset "([Account_000], [Project_000])"
  add values;
```

Export Data (Aggregate Storage)

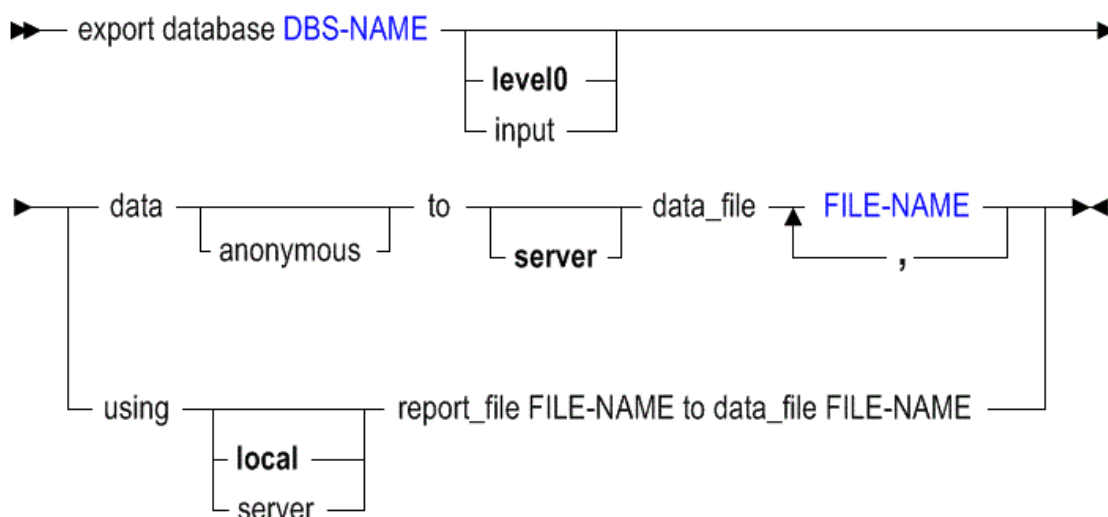
The MaxL **export data** statement helps you export data from an Essbase aggregate storage database.

[Click here for non-aggregate storage version](#)

Use this statement to export level-0 data, which does not include calculated values, from an aggregate storage cube. Export data files are written to the cube directory, unless an alternate path is specified using FILEGOVPATH configuration. To use Report Writer, export the data using a report file.

The minimum permission required to export data is Database Access.

Syntax



- **DBS-NAME**
- **FILE-NAME**

Keywords

You can export data from an aggregate storage database in the following ways using MaxL **export database**.

export database DBS-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 DBS-NAME input data...

This statement performs the same action as **export database DBS-NAME level0 data...**

export database DBS-NAME ... data anonymous

Export data in anonymized format. Anonymization removes the risk of sensitive data disclosure, and can be used in case sample data needs to be provided for technical support. Essbase replaces real data values with 1, for each value in the block.

Example:

```
export database ASOSamp.Basic data anonymous to data_file 'expobscured.txt';
```

export database DBS-NAME ... using ... report_file ...

Run a stored report script, exporting a subset of the database.

Example:

```
export database ASOSamp.Basic using server report_file 'A01' to data_file  
'REPORTa01.TXT';
```

Notes

- This statement requires the database to be started.
- 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, from 1 to 8 file names. This number should generally be equal to the number of processors on the machine that you wish to commit to doing parallel export. The number of threads Essbase uses typically depends on the number of file names you specify. However, on a very small aggregate storage database with a small number of data blocks, it is possible that only a single file will be created (in effect, performing serial export), 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`, and so on 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 `exportfile.txt`, the next additional file is `exportfile_1.txt`.

Example 1

The following example exports all level 0 data from ASOSamp.Basic to an export file.

```
export database ASOSamp.Basic data to server data_file 'myfilesamp.txt';
```

The export location is the cube directory (ASOSamp/Basic/myfilesamp.txt), or whichever Essbase Server directory the administrator has specified for FILEGOVPATH.

Example 2

The following example uses a report script, `Bottom.rep`, to export a subset of sorted data from `ASOSamp.Basic` to an output file, `Bottom.rpt`.

```
export database ASOSamp.Basic using report_file 'Bottom.rep' to data_file
'Bottom.rpt';
```

Sample Report Script and Output

For example 2, assume that `Bottom.rep` is the following report script file based on `ASOSamp.Basic`:

```
//Bottom.rep
<Sym
<Column (Measures, Years)
<Row (Geography, Products)
<ICHILDREN Geography
<ICHILDREN Products
<Bottom (3, @DataColumn(1))
!
```

The report script produces the following report (`Bottom.rpt`):

Measures	Years	Time	Transaction	Type	Payment	Type	Promotions	Age	Income	Level
Stores										
North East			All Merchandise						43,250,241	
			Products						43,250,241	
			High End Mercha~						11,379,402	
South			All Merchandise						32,790,838	
			Products						32,790,838	
			High End Mercha~						8,436,598	
Geography			All Merchandise						76,041,079	
			Products						76,041,079	
			High End Mercha~						19,816,000	

Export Query Tracking (Aggregate Storage)

The MaxL `export query_tracking` statement helps you export query tracking data from an Essbase aggregate storage (ASO) cube to a text file.

Query tracking captures user retrieval statistics against an ASO cube, enabling Essbase to make view-based optimizations to improve the performance of aggregations.

When an ASO cube is refreshed or restarted, query data is not persisted in the cube. As an optimization technique, before refreshing or restarting, you can export query tracking data to a text file. To rebuild aggregate views after a refresh or restart, import the query tracking data

from the text file. Essbase uses the query data to select the most appropriate set of aggregate views to materialize.

Before exporting query tracking data, query tracking must be enabled and working.

Query tracking is enabled by default for ASO cubes. To check whether it's enabled, you can confirm by running the following MaxL statement (example is for ASOSamp Basic cube):

```
query database ASOSamp.Basic get cube_size_info;
```

If you need to enable query tracking, use the [alter database](#) (aggregate storage) statement with the **enable query_tracking** grammar.

Syntax

```
▶▶ export query_tracking DBS-NAME to server file FILE-NAME ▶▶
```

- DBS-NAME
- FILE-NAME

Keywords

You can use MaxL **export query_tracking** to export ASO query tracking information in the following ways. The minimum application permission required to export query tracking data is Database Manager. Do not edit the text file with the exported query data.

export query_tracking DBS-NAME to server file...

Export query data from the specified aggregate storage cube to the specified file. Unless the administrator has specified a different export location, the file is created in the *<Application Directory>/app/appname/dbname* folder. If you do not know where *<Application Directory>* is in your environment, refer to Environment Locations in the Essbase Platform.

export query_tracking DBS-NAME to file...

You can omit the **server** keyword, but the result is the same.

 **Note:**

Query tracking and query tracing are different.

Query *tracking* enables you to capture user retrieval statistics against an aggregate storage cube, so that Essbase can make view-based optimizations to improve the performance of aggregations. It is on by default. Related MaxL statements include:

```
import query_tracking
export query_tracking
alter database enable query_tracking
query database appname.dbname get cube_size_info
```

Query *tracing* helps you monitor Essbase query performance metrics for block storage cubes (including hybrid mode). It is off by default. If you enable it, Essbase logs metrics in a trace report. Related configuration parameters: TRACE_REPORT, QUERYTRACE, QUERYTRACETHRESHOLD, LONGQUERYTIMETHRESHOLD.

Example

```
export query_tracking ASOsamp.Basic to server file
'query_data_aso_sample.txt';
```

Exports query data from the ASOsamp.Basic database to the named file. The export location is the cube directory (ASOsamp/Basic/), or whichever Essbase Server directory the administrator has specified for FILEGOVPATH.

See Also

[Import Query Tracking \(Aggregate Storage\)](#)

Import Data (Aggregate Storage)

The MaxL import data statement for ASO mode helps you load data into an Essbase aggregate storage database. The minimum application permission required to load data or dimensions is Database Update. Use this statement to import data from text files or other sources, with or without a rules file.

[Click here for non-aggregate storage version](#)

Syntax

You can load data or dimensions to an aggregate storage database in the following ways using MaxL **import data**.

Keywords

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.

import database DBS-NAME ...using ... rules_file

Import data into the database using a specified rules file. A separate rule file is required for each unique, non-Essbase source of data. If you're only re-importing data from native Essbase export files, you may not need to use a rule.

import database DBS-NAME ... 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.

import database DBS-NAME ... from data_string ...

Load a single data record into the selected database. The string following **data_string** must be a contiguous line, without newline characters.

Example:

```
import database ASO_TypedMeasures.Basic data from data_string '"Color"
"Shoes" "Connecticut" "Q1" #Txt:Empty' on error abort;
```

import database DBS-NAME ... connect as ...

If you are importing from an SQL source, you must always use a rule file. Provide the appropriate user name and password:

- If the network connectivity to the source data is saved in an Essbase connection and Datasource, provide your Essbase credentials in the MaxL statement. For example:

```
import database Sample.Basic data connect as "Essbaseadmin" identified by
"Essbasepa55w0RD" using server rules_file "myrulefile" on error write to
'loadds.err';
```

- Otherwise, provide the user name and password required to connect to the external RDBMS source. For example:

```
import database Sample.Basic data connect as "RDBMSuser" identified by
"RDBMSpa55w0RD" using server rules_file "myrulefile" on error write to
'loadds.err';
```

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 <buffer-block-spec>. 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.

Each rules file must use the same authentication information (SQL user name and password). In the following example, SQL data is loaded from two rules files (rule1.rul and rule2.rul):

```
import database ASOsamp.Basic 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";
```

In specifying the list of rules files, use a comma-separated string of rules file names (excluding the `.rul` extension). The file name for rules files must not exceed eight bytes and the rules files must reside on Essbase Server.

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 rule1) and increments the ID number by one for each subsequent data load buffer (for example, ID 101 for rule2). The ODBC driver you are using must be configured for parallel SQL connections.

Note:

Performing multiple SQL data loads in parallel to aggregate storage databases is different than using the **to load_buffer with buffer_id** grammar to load data into a buffer, and then using the **from load_buffer with buffer_id** grammar to explicitly commit the buffer contents to the database.

import database DBS-NAME ... to load_buffer with buffer_id ...

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.

import database DBS-NAME ... from load_buffer with buffer_id ...

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.

import database DBS-NAME ... from load_buffer with buffer_id ... override|add|subtract 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.

import database DBS-NAME ... 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.

import database DBS-NAME ... 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.

import database DBS-NAME ... 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.

Examples

The following example performs a data load using a data file stored in the shared folder of the Essbase file catalog. No rule file is needed.

```
import database 'ASOSamp'.'Basic' data from server data_file 'catalog/shared/  
ASO_Sample_Data' on error write to "asodataload.err";
```

The following example commits the contents of a specified data load buffer to the ASOSamp.Basic database.

```
import database ASOSamp.Basic data from load_buffer with buffer_id 1;
```

The following example commits the contents of multiple data load buffers (buffer_id 1 and buffer_id 2) to the ASOSamp.Basic database.

```
import database ASOSamp.Basic data from load_buffer with buffer_id 1, 2;
```

The following example commits the contents of a specified data load buffer to the ASOSamp.Basic database by adding values.

```
import database ASOSamp.Basic data from load_buffer with buffer_id 1 add  
values;
```

The following example commits the contents of the specified data load buffer into a new data slice in the ASOSamp.Basic database.

```
import database ASOSamp.Basic data from load_buffer with buffer_id 1 override  
values create slice;
```

The following example replaces the contents of the ASOSamp.Basic database with the contents of the specified data load buffer.

```
import database ASOSamp.Basic data from load_buffer with buffer_id 1 override  
all data;
```

The following example replaces the contents of all incremental data slices in the ASOSamp.Basic 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.

```
import database ASOSamp.Basic data from load_buffer with buffer_id 1 override  
incremental data;
```

For a full workflow of examples, refer to [Load Data Using Buffers](#).

Import Query Tracking (Aggregate Storage)

The MaxL `import query_tracking` statement helps you import query tracking data from a text file to an Essbase aggregate storage (ASO) cube.

Query tracking captures user retrieval statistics against an ASO cube, enabling Essbase to make view-based optimizations to improve the performance of aggregations.

When an ASO cube is refreshed or restarted, query data is not persisted in the cube. As an optimization technique, before refreshing or restarting, you can export query tracking data to a text file. To rebuild aggregate views after a refresh or restart, import the query tracking data from the text file. Essbase uses the query data to select the most appropriate set of aggregate views to materialize.

Keep in mind the following prerequisites before importing query tracking data:

- Query tracking must be enabled and working for an ASO cube.

Note:

Query tracking is enabled by default for aggregate storage cubes. To check whether it's enabled, you can confirm by running the following MaxL statement (example is for ASOSamp Basic cube):

```
query database ASOSamp.Basic get cube_size_info;
```

If you need to enable query tracking, use the [alter database](#) (aggregate storage) statement with the **enable query_tracking** grammar.

- Query tracking data for the cube must have been exported to a text file on the Essbase Server. Do not edit the text file with the exported query data.

Syntax

```

▶▶ import query_tracking DBS-NAME from server file FILE-NAME ▶▶

```

- DBS-NAME
- FILE-NAME

Keywords

You can use MaxL **import query_tracking** to import ASO query tracking information the following ways. The minimum application permission required to import query tracking data is Database Manager.

import query_tracking DBS-NAME from server file...

Import query data to the specified aggregate storage cube from the specified file. For FILE-NAME, specify the name of the text file that contains the query data to import. By default, the

file is created in the cube directory, or whichever Essbase Server directory the administrator has specified for FILEGOVPATH.

import query_tracking DBS-NAME from file...

You can omit the **server** keyword, but the result is the same.

Note:

Query tracking and query tracing are different.

Query *tracking* enables you to capture user retrieval statistics against an aggregate storage cube, so that Essbase can make view-based optimizations to improve the performance of aggregations. It is on by default. Related MaxL statements include:

```
import query_tracking
export query_tracking
alter database enable query_tracking
query database appname.dbname get cube_size_info
```

Query *tracing* helps you monitor Essbase query performance metrics for block storage cubes (including hybrid mode). It is off by default. If you enable it, Essbase logs metrics in a trace report. Related configuration parameters: TRACE_REPORT, QUERYTRACE, QUERYTRACETHRESHOLD, LONGQUERYTIMETHRESHOLD.

Example

```
import query_tracking ASOsamp.Basic from server file
'query_data_aso_sample.txt';
```

Imports the query data from the named file in the cube directory (or whichever directory the administrator has specified for export files) to the ASOsamp.Basic cube.

See Also

[Export Query Tracking \(Aggregate Storage\)](#)

Query Application (Aggregate Storage)

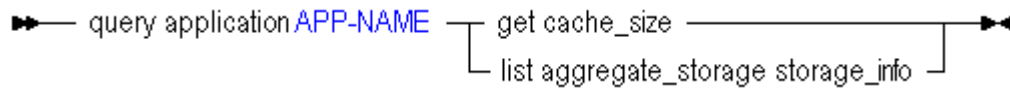
The MaxL **query application** statement helps you get information about the current state of an Essbase aggregate storage (ASO) application.

[Click here for block storage version](#)

This version of **query application** is only applicable for aggregate storage applications.

This statement requires the application to be started.

Syntax



APP-NAME

Examples

The following MaxL statement returns the maximum size (in kilobytes) to which the aggregate storage cache may grow:

```
query application ASOSamp get cache_size;
```

The following MaxL statement:

```
query application ASOSamp list aggregate_storage storage_info;
```

returns the following details about the ASO cache, I/O, and disk space allocation:

Table 3-10 Query Application List Aggregate Storage Storage_Info MaxL Output Columns

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.
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.


 **Note:**
This statistic may not be accurate when parallel data load or parallel calculation operations are in use.

Table 3-10 (Cont.) Query Application List Aggregate Storage Storage_Info MaxL Output Columns

Output Columns	Description
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).

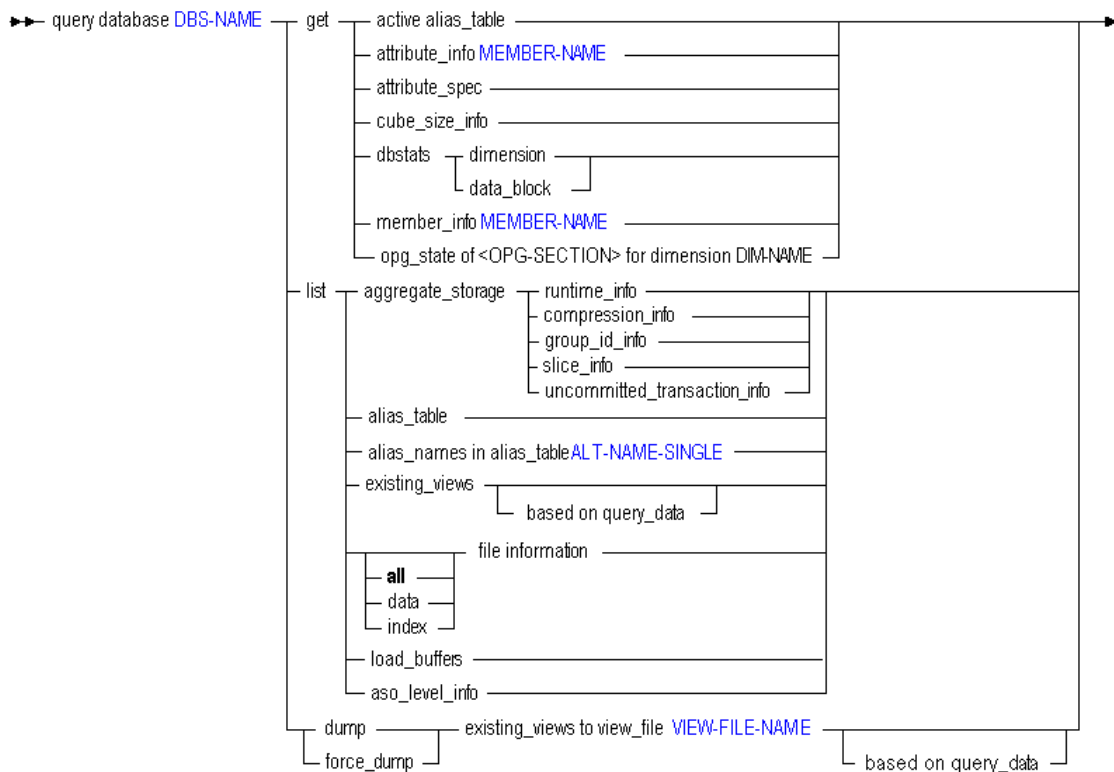
Query Database (Aggregate Storage)

The MaxL **query database** statement for ASO mode helps you retrieve advanced information about the current state of an Essbase aggregate storage cube that is running.

[Click here for non-aggregate storage version](#)

This statement requires the database to be started.

Syntax



- DBS-NAME
- MEMBER-NAME
- ALT-NAME-SINGLE
- VIEW-FILE-NAME

Keywords

You can query for active ASO database information in the following ways using MaxL **query database**. The minimum application permission required for most **query database** actions is Database Access, with exceptions noted.

query database DBS-NAME get active alias_table

Display the active alias table for the user issuing the statement.

Example:

```
query database ASOSamp.Basic get active alias_table;
```

query database DBS-NAME get attribute_info MEMBER-NAME

Get attribute member, dimension, and name information for the specified ASO attribute member.

Example:

```
query database ASOSamp.Basic get attribute_info 'Area Code';
```

query database DBS-NAME 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.

Example:

```
query database ASOSamp.Basic get attribute_spec;
```

query database DBS-NAME get cube_size_info

Display information about input data size, aggregated data size, and number of queries tracked (when query tracking is enabled).

Example:

```
query database ASOSamp.Basic get cube_size_info;
```

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.

Column Name	Contents
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> list existing_views • execute aggregate process • execute aggregate selection <p>Query tracking is on by default.</p>

query database DBS-NAME get dbstats dimension

Get information about dimensions. This action requires Database Access permission or higher.

Example:

```
query database ASOsamp.basic get dbstats dimension;
```

The **index_type** field values are numeric, and translate as follows:

0	Dense
1	Sparse
3	None (database is aggregate storage)

query database DBS-NAME get dbstats data_block

Get information about data blocks. This action requires Database Access permission or higher. The information returned has little relevance to aggregate storage databases.

Example:

```
query database ASOsamp.basic get dbstats data_block;
```

query database DBS-NAME get member_info MEMBER-NAME

Get information on a specific member. This action requires Database Access permission or higher.

Example:

```
query database ASOsamp.basic get member_info 'Original Price';
```

Output Columns for Member Info

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

query database DBS-NAME 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).

Example:

```
query database ASOsamp.basic get opg_state of member_data for dimension
'Geography';
```

This action requires Database Access permission or higher. Refer to [Outline Paging Dimension Statistics](#) for a description of the output.

query database DBS-NAME get opg_state of member_name_namespace ...

Display information that matches member names to internal member identifiers (one section per database, thus the information for all dimensions is the same).

Example:

```
query database ASOsamp.basic get opg_state of member_name_namespace for
dimension 'Products';
```

This action requires Database Access permission or higher. Refer to [Outline Paging Dimension Statistics](#) for a description of the output.

query database DBS-NAME get opg_state of member_formula ...

Display all formulas for the dimension.

Example:

```
query database ASOsamp.basic get opg_state of member_formula for dimension
'Measures';
```

This action requires Database Access permission or higher. Refer to [Outline Paging Dimension Statistics](#) for a description of the output.

query database DBS-NAME get opg_state of member_UDA ...

Display all user defined attributes (UDAs) for the dimension.

Example:

```
query database ASOsamp.basic get opg_state of member_uda for dimension
'Measures';
```

This action requires Database Access permission or higher. Refer to [Outline Paging Dimension Statistics](#) for a description of the output.

query database DBS-NAME get opg_state of member_UDA_namespace ...

Display information that matches UDAs to internal member identifiers.

Example:

```
query database ASOsamp.basic get opg_state of member_uda_namespace for
dimension 'Measures';
```

This action requires Database Access permission or higher. Refer to [Outline Paging Dimension Statistics](#) for a description of the output.

query database DBS-NAME get opg_state of attribute_to_base_member_association ...

Display information that identifies the attribute member associated with each base member of the dimension.

Example:

```
query database ASOsamp.basic get opg_state of
attribute_to_base_member_association for dimension 'Stores';
```

This action requires Database Access permission or higher. Refer to [Outline Paging Dimension Statistics](#) for a description of the output.

query database DBS-NAME get opg_state of member_comment

Display all member comments for the dimension.

Example:

```
query database ASOsamp.basic get opg_state of member_comment for dimension
'Age';
```

This action requires Database Access permission or higher. Refer to [Outline Paging Dimension Statistics](#) for a description of the output.

query database DBS-NAME get opg_state of member_alias_namespace

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).

Example:

```
query database ASOsamp.basic get opg_state of member_alias_namespace for
dimension 'Age';
```

This action requires Database Access permission or higher. Refer to [Outline Paging Dimension Statistics](#) for a description of the output.

query database DBS-NAME list aggregate_storage runtime_info

Display runtime statistics about the aggregate storage database.

Example:

```
query database ASOsamp.basic list aggregate_storage runtime_info;
```

This action requires Database Access permission or higher. For a description of the output returned by this statement, see [Aggregate Storage Runtime Statistics](#).

query database DBS-NAME list aggregate_storage group_id_info

Display information about group IDs and their timestamps related to General Ledger cubes.

 **Note:**

This grammar applies to General Ledger cubes, not to non-general-ledger aggregate storage databases. For normal aggregate storage databases, this table will be empty.

This MaxL grammar is disabled for previous release Essbase MaxL clients.

Example:

```
query database ASOsamp.basic list aggregate_storage group_id_info;
```

This statement returns the following output:

Column Name	Contents
group_id	The allocation group id, according to the begin allocation command that created the allocation group. The number is an unsigned 64-bit integer.

Column Name	Contents
transaction_id	The aggregate storage transaction ID that is used internally. The number is an unsigned 64-bit integer.
state	A string describing the state of the group ID. For example: BeginAllocation Done, Allocation In Progress, Allocation Done, EndAllocation In Progress.
time_last_used	The date and time the group ID was last used. The value is either the time the group ID was created or the time that an allocation or custom calculation was last performed with this group ID. The value is a string.
time_expired	The date and time when the group ID will time out (expire). The value is a string.
expired	Indicates whether the group ID has timed out. If the group ID has expired, the group ID will be rolled back the next time a begin allocation command is executed. The value is a boolean.

This action requires Database Access permission or higher. For a description of the output returned by this statement, see [Aggregate Storage Group ID Information Output](#).

query database DBS-NAME list aggregate_storage slice_info

Display information about data slices and views, some information of which applies only to General Ledger cubes (not to non-general-ledger aggregate storage databases).

Note:

Small incremental slices may have fewer aggregate views than the primary slice (slice number 0). Incremental slices with less than 100,000 cells will never have any aggregate views built. However, if an incremental slice is larger than 100,000 cells and it is larger than the primary slice, then it will always have the same aggregate views as the primary slice.

Example:

```
query database ASOsamp.basic list aggregate_storage slice_info;
```

This MaxL grammar is disabled for previous release Essbase MaxL clients.

This statement returns the following output:

Column Name	Contents
transaction_id	<p>(Applies to General Ledger cubes only) The ID of the transaction to which this slice and view belong. There is one transaction ID for each GL group ID. The number is an unsigned 64-bit integer. To find the corresponding group ID, use the following MaxL command:</p> <pre>query database app.db list aggregate_storage group_id_info;</pre> <p>For non-general-ledger aggregate storage databases, this number is always 0.</p>
slice_id	<p>ID number of the data slice. The number is an unsigned 32-bit integer.</p>
slice_tag	<p>(Applies to General Ledger cubes only) When an allocation or custom calculation is done within an allocation begin/end, this number is the rule_id of the allocation that made this data slice. The number is an unsigned 64-bit integer. For non-general-ledger aggregate storage databases, this number is always 0.</p>
view_id	<p>0 indicates an input view; otherwise, the view is an aggregate view. The number is an unsigned 64-bit integer. To list the levels in a given aggregate view, use the following MaxL command:</p> <pre>query database app.db list existing_views;</pre>
size_cells	<p>The number of cells in the given view of the slice. The number is an unsigned 64-bit integer.</p>
size_kb	<p>The size in KB of the given view of the slice. The number is an unsigned 64-bit integer.</p>

This action requires Database Access permission or higher. For a description of the output returned by this statement, see [Aggregate Storage Slice Information Output](#).

query database DBS-NAME list aggregate_storage uncommitted_transaction_info

Display information about uncommitted transactions that are related to General Ledger cubes.

Example:

```
query database ASOsamp.basic list aggregate_storage
uncommitted_transaction_info;
```

 **Note:**

This grammar applies to General Ledger cubes, not to non-general-ledger aggregate storage databases. For normal aggregate storage databases, this table will be empty.

This MaxL grammar is disabled for previous release Essbase MaxL clients.

This action requires Database Access permission or higher. This statement returns the following output:

Column Name	Description
unc_transactions	The number of existing user transactions that are not yet committed.
unc_data_slices	The number of data slices used by uncommitted transactions.
unc_input_data_size_cells	The number of input cells used by uncommitted transactions.
unc_aggregate_views	The number of aggregate views used by uncommitted transactions.
unc_aggregate_data_size_cells	The number of aggregate cells used by uncommitted transactions.
unc_input_data_size_kb	The total disk space used by uncommitted input-level data.
unc_aggregate_data_size_kb	The total disk space occupied by uncommitted aggregate cells.

This action requires Database Access permission or higher. For a description of the output returned by this statement, see [Aggregate Storage Uncommitted Transaction Information Output](#).

query database DBS-NAME list aggregate_storage compression_info

Display estimated compression for aggregate storage databases when different dimensions are hypothetically used as the compression dimension. These estimates can help you choose the best dimension to use as the compression dimension.

In aggregate storage databases, the compression dimension enables database compression. A good candidate for a compression dimension is one that optimizes data compression and maintains retrieval performance. The following table lists data for all non-attribute dimensions, even though it may not be possible to select them as the compression dimension without significant changes to the outline. For information on the requirements of a compression dimension, see [Understanding the Compression Dimension for Aggregate Storage Databases](#).

Example:

```
query database ASOsamp.basic list aggregate_storage compression_info;
```

This action requires Database Access permission or higher. This statement returns the following output:

Column Name	Contents
dimension_name	Each dimension name in the database, hypothetically considered to be the compression dimension.
is_compression	Indicates whether the dimension is the aggregate storage compression dimension. (There can be only one compression dimension in an aggregate storage database.)
stored_level0_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 a compression dimension.
average_bundle_fill	Estimated average number of values per compression dimension bundle. Choosing a compression 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.
level0_mb	Estimated size of the compressed database, in megabytes. A smaller expected level-0 size indicates that choosing this dimension enables better compression. Except for the scenario in which there is no compression dimension (<i>None</i>), 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 <i>None</i> .

query database DBS-NAME list alias_table

Get a list of alias tables that are defined for the database. This action requires Database Access permission or higher.

Example:

```
query database ASOsamp.Basic list alias_table;
```

query database DBS-NAME 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. This action requires Database Access permission or higher.

Example:

```
query database ASOsamp.Basic list alias_table;
```

query database DBS-NAME list existing_views

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.

This action requires Database Access permission or higher.

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.

To use the **based on query_data** clause, query tracking must be enabled. Query tracking is enabled by default, but if you need to enable it, use [alter database <db-name> enable query tracking](#).

Example:

```
query database ASOsamp.Basic list existing_views based on query_data;
```

query database DBS-NAME 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.

Example:

```
query database ASOsamp.Basic list all file information;
```

The above statement returns:

file_name	file_type	file_number	number_of_type
file_size_kb	file_size_bytes	opened	
Index File Total		1	0
0	0	0	FALSE
Data File Total		2	0
0	0	0	FALSE
Grand File Total		3	0
0	0	0	FALSE

This action requires Database Access permission or higher.

query database DBS-NAME list load_buffers

Display a list and description of the data load buffers that exist on an aggregate storage database.

Example:

```
query database ASOsamp.Basic list load_buffers;
```

This action requires Database Manager permission or higher.

Refer also to [List Aggregate Storage Data Load Buffers](#)

query database DBS-NAME list aso_level_info

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).

This action requires Database Manager permission or higher.

Example:

```
query database ASOsamp.Basic list aso_level_info;
```

This statement returns:

dimension	num_levels
Measures	1
Years	1
Time	4
Transaction Type	2
Payment Type	2
Promotions	2
Age	3
Income Level	2
Products	6
Stores	7
Geography	7

query database DBS-NAME dump|force_dump existing_views...

Saves existing views of this ASO database to an aggregation script. This action requires at least Database Access permission.

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.

If the **based on query_data** phrase is used, the view selection that is saved will be based on previously collected query-tracking data. For more information about query tracking, see the **based on query_data** description in [execute aggregate selection](#).

Example:

```
query database ASOsamp.Basic dump existing_views to view_file viewdump;
```

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.

Table 3-16 Outline Paging Dimension Statistics MaxL Output Columns

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.

Table 3-16 (Cont.) Outline Paging Dimension Statistics MaxL Output Columns

Column Name	Contents
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

You can examine the runtime statistics about a currently active Essbase aggregate storage (ASO) cube to learn how many stored levels each dimension has, as well as other information that can help you reduce the size of the cube.

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 cubes; 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 cube. The number of bits each dimension uses in the dimensional key is shown in the value column for each dimension.

The number of 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 the key length be 8 bytes instead of 16, an improvement which reduces the overall size of the cube. 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 Cube

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)                       8
Number of input-level cells                   0
Number of incremental data slices             0
Number of incremental input cells             0
Number of aggregate views                     0
Number of aggregate cells                     0
Number of incremental aggregate cells         0
Cost of querying incr. data (ratio to total cost) 0
Input-level data size (KB)                   0
Aggregate data size (KB)                     0
```

The whole-cube statistics are described in the following table.

Table 3-17 Aggregate Storage Runtime Statistics MaxL Output

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 cube, including incremental slices.
Number of incremental data slices	The number of data slices resulting from incremental data loads.

Table 3-17 (Cont.) Aggregate Storage Runtime Statistics MaxL Output

Column Name	Description
Number of incremental input cells	The number of level-0 cells in the incremental data slices. To see the number of unique aggregate views, use the MaxL statement: <code>query database appname.dbname list existing_views;</code>
Number of aggregate views	The number of aggregate views in the cube, including those automatically built on incremental slices.
Number of aggregate cells	The number of cells stored in the cube's aggregate views.
Number of incremental aggregate cells	The number of cells stored in the incremental slices' aggregate views.
Cost of querying incr. data (ratio to total cost)	The average percentage of query time spent processing incremental data slices. This functionality is useful in deciding when slices should be merged together to improve query performance.
Input-level data size (KB)	The total disk space used by input-level data.
Aggregate data size (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.

Aggregate Storage Slice Information Output

The following MaxL statement:

```
query database "dmglex4"."basic" list aggregate_storage slice_info;
```

Returns the following output:

```

transaction_id  slice_id  slice_tag  view_id  size_cells  size_kb
-----+-----+-----+-----+-----+-----
                0         0         0         0         38        64
                3         1        66         0         21        32
                3         2        77         0         21        32

```

See [Query Database](#).

Aggregate Storage Group ID Information Output

The following MaxL statement:

```
query database "dmglex4"."basic" list aggregate_storage group_id_info;
```

Returns the following output:

```

group_id transaction_id state time_last_used
time_expired expired
+-----+-----+-----+-----+
|      1234      |      1 | Allocation Done | Wed Jul 20 17:39:57 | Wed Jul 20 |
| 17:44:57      | FALSE |                  |                      |              |
+-----+-----+-----+-----+

```

See [Query Database](#).

Aggregate Storage Uncommitted Transaction Information Output

The following MaxL statement:

```
query database "dmglex4"."basic" list aggregate_storage
uncommitted_transaction_info;
```

Returns the following output (columns are truncated):

```

unc_trans unc_data_ unc_input unc_aggre unc_aggre unc_input unc_aggre
+-----+-----+-----+-----+-----+-----+-----+
|      0      |      0      |      0      |      0      |      0      |      0      |      0      |
+-----+-----+-----+-----+-----+-----+-----+

```

See [Query Database](#).

MaxL Definitions

To become an advanced user of MaxL, explore this documentation to learn about the parts of MaxL statements, MaxL's syntax and quoting rules, and the Essbase data types that comprise MaxL statement terminals.

This section contains the following topics:

- [MaxL Syntax Notes](#)
- [Numbers in MaxL Syntax](#)
- [Terminals](#)
- [Privileges and Roles](#)
- [Quoting and Special Characters Rules for MaxL Language](#)

MaxL Syntax Notes

MaxL syntax is comprised of statements, tokens, keywords, terminals names, strings, numbers, and delimiters.

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 external 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 a string which can be quoted or unquoted. Unquoted names must begin with an alphabetic character. Quoted names can consist of any sequence of characters. 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](#).

The MaxL **alphabet** consists of the following elements:

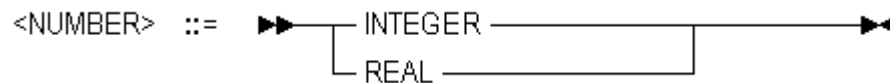
Table 3-18 MaxL Alphabet Elements

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).
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

Numbers in MaxL Syntax

Numbers in the MaxL language are either integers or reals. Numbers are important elements of MaxL statements used by administrators to work with Essbase.

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

MaxL *terminals* are discrete parts of the language that cannot be broken down further. Usually, a MaxL terminal is a common Essbase data type. Terminals are important elements of MaxL statements used by administrators to work with Essbase.

The following section lists terminals in alphabetical order.

- [ACTION](#)
- [ALLOC-NUMERIC](#)
- [ALT-NAME-SINGLE](#)
- [APP-NAME](#)
- [AREA-ALIAS](#)
- [BUFFER-ID](#)
- [CALC-NAME](#)
- [CALC-NAME-SINGLE](#)
- [CALC-SPEC-STRING](#)
- [CALC-STRING](#)

- COLUMN-WIDTH
- COMMENT-STRING
- CONDITION
- CUBE-AREA or MDX-SET
- DATE
- DBS-EXPORT-DIR
- DBS-NAME
- DBS-STRING
- DIM-NAME
- EXPORT-DIR
- FILE-NAME
- FILE-NAME-PREFIX
- FILTER-NAME
- FULL-EXPORT-DIR
- FUNC-NAME
- GROUP-NAME
- HOST-NAME
- ID-RANGE
- ID-STRING
- IMP-FILE
- IMPORT-DIR
- JAVACLASS.METHOD
- LOCATION-ALIAS-NAME
- LOC-ALIAS-SINGLE
- LOG-TIME
- MACRO-EXPANSION
- MACRO-NAME
- MEMBER-NAME
- OBJ-NAME
- OBJ-NAME-SINGLE
- OUTLINE-ID
- PASSWORD
- PATHNAME_FILENAME
- PRECISION-DIGITS
- PROPS
- RNUM
- RTSV-LIST
- RULE-FILE-NAME

- [SESSION-ID](#)
- [SIZE-STRING](#)
- [SPOOL-NAME](#)
- [STOPPING-VAL](#)
- [TABLSP-NAME](#)
- [TRIGGER-NAME](#)
- [URL-NAME](#)
- [USER-NAME](#)
- [VARIABLE-NAME](#)
- [VIEW-FILE-NAME](#)
- [VIEW-ID](#)
- [VIEW-SIZE](#)

ACTION

The ACTION terminal in MaxL represents the required action if an Essbase data-monitoring trigger is activated.

Syntax

```
mail [smtp],[sender],[receiver1,receiver2,...],[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 cubes.
- spool - logs a message in a specified file in the \trig folder under the cube directory.

Type

string (see [MaxL Syntax Notes](#))

Example

```
mail manager.sales.com, [mktdir@CC.com, Monitor@acnts.com]
spool "trgmonitor"
```

Referenced By

[create trigger](#)

[drop trigger](#)

ALLOC-NUMERIC

The ALLOC-NUMERIC terminal in the MaxL language for Essbase is an MDX numeric value expression used to specify the amount value for an ASO custom allocation source.

The amount value is allocated to cells in the target region of an Essbase aggregate storage (ASO) cube. 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](#))

Examples

- (Acc_1000, Jan_2009)
- 100.00
- (Acc_1000 + Acc_2000) / 2
- AcctA + AcctB
- Balance * 1.1

Referenced By

[execute allocation](#)

ALT-NAME-SINGLE

The ALT-NAME-SINGLE terminal in the MaxL language for Essbase represents the name of an alias table.

If the name contains special characters (see [MaxL Syntax Notes](#)), it must be enclosed in single or double quotation marks when you pass it into the MaxL Shell.

Type

name (see [MaxL Syntax Notes](#))

Example

```
Region  
'Long Names'
```

Referenced By

[alter database](#)

[query database](#)

APP-NAME

The APP-NAME terminal in MaxL represents the name of an Essbase application.

The application name must not exceed 8 bytes (non-Unicode-mode applications) or 30 characters (Unicode-mode applications). Avoid using spaces. Application names are not case-sensitive.

If the name contains any allowed special characters, it must be enclosed in single or double quotation marks when you pass it into the MaxL Shell. 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](#))

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](#)
[query application](#)
[refresh custom definitions](#)

AREA-ALIAS

The AREA-ALIAS terminal in the MaxL language is a shortcut name that you can associate with a specified cube area. The cube area is required to create an Essbase database partition. The area alias is optional.

An area alias is a shorthand name used 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](#))

Example

In the create partition statement below, "myalias" 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.

```
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) ' myalias
    mapped myalias (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

The BUFFER-ID terminal in the MaxL language is an identifier for an Essbase aggregate storage (ASO) data load buffer.

The load buffer ID must be a number between 1 and 999,999 inclusive. To destroy a load buffer before the incremental data load is complete, you must use the same BUFFER-ID number that was used to initialize it.

Type

number (see [MaxL Syntax Notes](#))

Referenced By

[alter database \(aggregate storage\)](#)

CALC-NAME

The CALC-NAME terminal in the MaxL language for Essbase represents the name of a stored calculation.

Syntax

- **Syntax for database-level calculation:**

name1.name2.name3

- **Syntax for application-level calculation:**

name1.name3

- *name1*—Application name.

- *name2*—Database name.
- *name3*—Calculation script name.

Type

name (see [MaxL Syntax Notes](#))

For database-level calculations, three tokens are required: two to indicate application and database context, and one as the calculation name.

Example

```
Sample.basic.'alloc'
```

For application-level calculations, two tokens are required: one to indicate application context, and one as the calculation name. When executing application-level calculations, you must specify which database to calculate using the syntax 'on database STRING.'

Example

- `Sample.'alloc'` is the application-level CALC-NAME.
- `execute calculation Sample.'alloc' on database Basic;` is a way to execute the application-level calculation on a database.

If any part of the name contains special characters (as listed in [MaxL Syntax Notes](#)), it must be enclosed in single or double quotation marks.

Referenced By

[create calculation](#)

[display calculation](#)

[drop calculation](#)

[execute calculation](#)

[grant](#)

CALC-NAME-SINGLE

The CALC-NAME-SINGLE terminal in the MaxL language for Essbase represents the short name of a stored calculation.

The CALC-NAME-SINGLE is the third token of a database-level [CALC-NAME](#) given to a stored Essbase calculation; in other words, the name of the calc script file, without the file extension.

If any part of the name contains special characters (see [MaxL Syntax Notes](#)), it must be enclosed in single or double quotation marks when you pass it into the MaxL Shell.

Type

name (see [MaxL Syntax Notes](#))

Example

If the full database-level calc name is `sample.basic.'alloc'`, then CALC-NAME-SINGLE is `'alloc'`.

Referenced By[alter database \(set\)](#)

CALC-SPEC-STRING

The CALC-SPEC-STRING terminal in the MaxL language is an optional string you can use to document the syntax of a custom-defined function or macro you have added to the Essbase calculator library.

This calculator-syntax specification string must be enclosed in single quotation marks when you pass it into the MaxL Shell.

Type

string (see [MaxL Syntax Notes](#))

Example

```
'@COVARIANCE (expList1, expList2)'
```

Use CALC-SPEC-STRING if the function or macro needs to be returned through the API that lists functions.

Referenced By[create function](#)[create macro](#)

CALC-STRING

The CALC-STRING terminal in the MaxL language represents the body of an anonymous (unstored) Essbase calculation, or the string used to specify the body of a stored calculation at the time you create it. It is a single token string.

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 when you pass it into the MaxL Shell.

Type

string (see [MaxL Syntax Notes](#))

Example

```
CALC DIM(Year, Measures, Product);
```

Referenced By[alter database \(set\)](#)[execute calculation](#)

COLUMN-WIDTH

The COLUMN-WIDTH terminal in the MaxL language for Essbase represents the width of the columns that should appear in MaxL display output tables.

The value for COLUMN-WIDTH must be a number (at least 8), or the keyword **default**. If you select **default**, the column width is 20 characters.

Type

number (see [MaxL Syntax Notes](#)) or **default**

Example

```
set display column width 80
set display column width default
```

Referenced By

[MaxL Shell Commands](#) (in Set Display Column Width)

COMMENT-STRING

The COMMENT-STRING terminal in the MaxL language for Essbase represents a string of descriptive / informational text you supply to describe an Essbase artifact.

If the comment string contains special characters (see [MaxL Syntax Notes](#)), it must be enclosed in single or double quotation marks when you pass it into the MaxL Shell.

Type

string (see [MaxL Syntax Notes](#))

Example

```
'This is a comment.'
```

Referenced By

[alter application](#)

[alter database \(set\)](#)

[alter database \(misc\)](#)

[alter database \(for ASO\)](#)

[create application](#)

[create database](#)

[create function](#)

[create macro](#)

[create partition](#)

CONDITION

The **CONDITION** terminal in the MaxL language represents a conditional expression serving as the test for whether an Essbase database trigger is activated.

The condition is a numeric-value-expression developed in MDX. Enclose strings containing spaces or other special characters in square brackets ([]).

Type

string (see [MaxL Syntax Notes](#))

Example

The following MaxL statement creates a trigger that logs information based on a condition. The **CUBE-AREA** region to track (after the **where** clause) and the **CONDITION** to test for (after the **when** clause) are expressed using MDX.

```
create or replace on update trigger Sample.Basic.NYCola where (Jan, Sales,
Actual, [100-10], [New York]) when [New York] > 20 then spool EastColas_Fail
end;
```

Referenced By

[create trigger](#)

CUBE-AREA or MDX-SET

The **CUBE-AREA** or **MDX-SET** terminal in the MaxL language represents an Essbase cube area, region, or other data set 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 MDX runtime functions such as **Filter**, **TopSum**, or **BottomSum**. Enclose strings containing spaces or special characters in square brackets ([]). Check the MaxL statement examples for other quoting requirements, which may differ depending on what you are doing.

Type

string (see [MaxL Syntax Notes](#))

Examples

The following is an MDX set of siblings.

```
'{[Jan 2000], [Feb 2000], [Mar 2000}]'
```

The following is an MDX crossjoined set.

```
'{([Qtr1], [New York]), ([Qtr1], [California]),
([Qtr2], [New York]), ([Qtr2], [California])}'
```

The following MDX set is also a tuple.

```
'{(Jun, FY2011, Actual)}'
```

The following MaxL statement clears data from a region of ASOsamp.Basic. The region to clear is expressed using MDX.

```
alter database ASOsamp.Basic clear data in region '{(Coupon, [Prev Year], South)}' physical;
```

The following MaxL statement creates a trigger that logs information based on a condition. The CUBE-AREA region to track (after the **where** clause) and the CONDITION to test for (after the **when** clause) are expressed using MDX.

```
create or replace on update trigger Sample.Basic.NYCola where (Jan, Sales, Actual, [100-10], [New York]) when [New York] > 20 then spool EastColas_Fail end;
```

Referenced By

[create trigger](#)
[alter database \(aggregate storage\)](#)
[execute allocation](#)
[execute calculation \(aggregate storage\)](#)

DATE

The DATE terminal in the MaxL language for Essbase is a validly formatted date string.

A date string must be formatted as follows:

- 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](#)), it must be enclosed in single or double quotation marks.

Type

string (see [MaxL Syntax Notes](#))

Example

```
'04/16/03'  
'04.16.2003'  
04_16_2003
```

Referenced By

[alter database \(misc\)](#)
[query database](#)

DBS-EXPORT-DIR

The DBS-EXPORT-DIR terminal in the MaxL language is the suffix of the name of an Essbase cube directory you specify, using the **export lro** statement, to contain export files for your database's linked reporting objects (LROs).

Essbase creates the LRO export directory in the application directory, with a full name such as `appname-dbname-suffix`.

After an LRO export is completed, the directory contains file-type LRO binary files (if applicable to the database), and the LRO-catalog export file with file-extension `.exp`.

Example: For a Sample.Basic export of LROs, if DBS-EXPORT-DIR is given as `lros`, then the `sample-basic-lros` directory is created in the application directory. 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 fails, as a safeguard against overwriting.

Type

string (see [MaxL Syntax Notes](#))

Referenced By

[export lro](#)

DBS-NAME

The DBS-NAME terminal in the MaxL language represents the name of an Essbase database (also known as a cube).

Two tokens are required in the name, to indicate application context.

Syntax

`name1.name2`

- *name1*—The name of the application containing the database.
The application name must not exceed 8 bytes (non-Unicode-mode applications) or 30 characters (Unicode-mode applications). Avoid using spaces.
- *name2*—The name of the database.
The database name must not exceed 8 bytes (non-Unicode-mode applications) or 30 characters (Unicode-mode applications). Avoid using spaces.

Database names are not case-sensitive.

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](#))

Example

```
Sample.basic
```

Referenced By

[alter database](#)

[alter database \(for ASO\)](#)

[alter partition](#)

[alter system](#)

[alter trigger](#)

[create database](#)

[create location alias](#)

[create outline](#)

[create partition](#)

[display database](#)

[display filter](#)

[display filter row](#)

[display location alias](#)

[display lock](#)

[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 DBS-STRING terminal in the MaxL language represents the second token of a DBS-NAME, which is name of an Essbase database (also known as a cube).

The DBS-STRING is the second token of [DBS-NAME](#). The limit for this token is 8 characters.

If the token contains special characters (see [MaxL Syntax Notes](#)), it must be enclosed in single or double quotation marks when you pass it into the MaxL Shell.

Type

string (see [MaxL Syntax Notes](#))

Example

```
basic
```

Referenced By

[alter application](#)
[alter application \(for ASO\)](#)
[alter database \(misc\)](#)
[alter database \(for ASO\)](#)
[alter partition](#)

[execute calculation](#)

DIM-NAME

The DIM-NAME terminal in MaxL represents the name of a dimension in an Essbase database outline.

If the dimension string contains special characters (see [MaxL Syntax Notes](#)), it must be enclosed in single or double quotation marks when you pass it into the MaxL Shell.

Type

string (see [MaxL Syntax Notes](#))

Example

```
Year  
Market
```

Referenced By

[query database](#)

EXPORT-DIR

The EXPORT-DIR terminal in the MaxL language represents the name of an Essbase cube directory to which linked reporting objects (LROs) were previously exported. This is used by the MaxL **alter system** statement when you need to remove the LRO export directory.

This directory is expected to exist under the application directory, and to contain LRO-catalog information exported using [Export LRO](#). Give only the directory name; do not give a full path. The directory specification must be enclosed in single or double quotation marks when you pass it into the MaxL Shell. The typical format is `appname-dbname-suffix`.

Type

string (see [MaxL Syntax Notes](#))

Example

```
'sample-basic-out'
```

Referenced By

[alter system](#) (delete export_directory)

FILE-NAME

The FILE-NAME terminal in the MaxL language represents a file name or path in Essbase.

If the FILE-NAME string contains special characters, 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](#))

Examples

- file01
- "errors.txt"
- '/Sample/Basic/expsamp.txt'
- 'D:\\filename'

Referenced By

[alter database \(misc\)](#)

[alter database \(for ASO\)](#)

[export data](#)

[import data](#)

[import dimensions](#)

FILE-NAME-PREFIX

The FILE-NAME-PREFIX terminal in the MaxL language represents the first part of a file name or path in Essbase. This prefix is used only by the MaxL **display drillthrough** statement.

If you use display drillthrough DBS-NAME to FILE-NAME-PREFIX on the client in the working directory of MaxL execution, URL drill through information is written to files generated with this prefix.

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](#)), it must be enclosed in single or double quotation marks.

Type

string (see [MaxL Syntax Notes](#))

Example

```
urlxmls
```

Referenced By

[display drillthrough](#)

FILTER-NAME

The FILTER-NAME terminal in the MaxL language represents the name of an Essbase security filter. The filter name in MaxL has three parts, in the form of `appname.dbname.filtername`.

Three tokens are required, to indicate application and database context.

The following special characters are not permitted:

```
! @ # $ % ^ & * ( ) _ + - = { } [ ] | ; ' : " < > ? , . / ~ `
```

Syntax

```
name1.name2.name3
```

- *name1*—Application name.
- *name2*—Database name.
- *name3*—Filter name.

Type

name (see [MaxL Syntax Notes](#))

Example

```
Sample.basic.filt1
```

Referenced By

[alter filter](#)

[create filter](#)

[display filter](#)

[display filter row](#)

[drop filter](#)

[grant](#)

FULL-EXPORT-DIR

The FULL-EXPORT-DIR terminal in the MaxL language is used by the **export lro** statement. It is a full path to an Essbase cube directory to contain export files for linked reporting objects (LROs).

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 `/scratch/exports/lros`, then the `lros` directory structure is created under `/scratch/exports/` if `/scratch/exports/` 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 `/scratch/exports` directory structure exists, MaxL creates the

`lros` directory as a subdirectory of `/scratch/exports`, but if `/scratch/exports` does not exist, MaxL will fail to create `/scratch/exports/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](#))

Examples

Windows

```
'C:\\temp\\lros'
```

Linux

```
'/scratch/exports/lros'
```

Referenced By

[export lro](#)

FUNC-NAME

The FUNC-NAME terminal in the MaxL language represents the name of a custom-defined Essbase calculator function (CDF). A global function name is a single token, while a local (application-level) function name has 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. Oracle recommends that you start a function name with `@`.

Any token of the name that contains special characters (see [MaxL Syntax Notes](#)), must be enclosed in single or double quotation marks.

Syntax

Syntax for local (application-level) function:

```
name1.name2
```

Syntax for global function:

```
name2
```

See [MaxL Syntax Notes](#)

- *name1*—Application name.
- *name2*—Function name.

Type

name (see [MaxL Syntax Notes](#))

Example

- Example of a local function:

```
Sample. '@COVARIANCE'
```

- Example of a global function:

```
'@COVARIANCE'
```

Referenced By

[create function](#)

[display function](#)

[drop function](#)

GROUP-NAME

The GROUP-NAME terminal in the MaxL language represents name of an Essbase security group.

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
- The following special characters are not permitted:

```
. ; , = + * ? [ ] | < > \ " ' / [Space] [Tab]
```

- If the group name contains any special characters (see [MaxL Syntax Notes](#)), the name must be enclosed in single or double quotation marks.

Types

- name (see [MaxL Syntax Notes](#))
- name@provider
- WITH IDENTITY [ID-STRING](#)

 **Note:**

If a user or group name includes the @ character, you must specify the provider as well. 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](#)

[create group](#)

[display privilege](#)

[drop group](#)

[grant](#)

HOST-NAME

The HOST-NAME terminal in MaxL represents the URL of the Essbase Server.

Use the discovery URL instead of a host name. A discovery URL is the URL provided by your Service Administrator, with `/agent` appended to the end. For example, `https://192.0.2.1:443/essbase/agent`.

Leading or trailing spaces will be trimmed off. Maximum length is 1024 bytes (non-Unicode application) or characters (Unicode application).

ID-RANGE

The ID-RANGE terminal in the MaxL language for Essbase represents a comma-separated list of sequence ID ranges for logged sequential transactions.

An transaction ID 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](#))

Example

```
1 to 10,20 to 100
```

Referenced By

[alter database \(misc\)](#)

ID-STRING

The ID-STRING terminal in the MaxL language for Essbase represents a unique attribute identifying a user or group in a directory.

Example

```
native://nvid=f0ed2a6d7fb07688:5a342200:1265973105c:-7f46?USER
```

Referenced By

[USER-NAME](#)

[GROUP-NAME](#)

IMP-FILE

The IMP-FILE terminal represents a data or rule file name or path used for Import statements in MaxL. The file location can be specified as **local** or **server**, where local means from the same filesystem where the statement is issued, and server means the Essbase Server file catalog.

If the data or rule file is specified to be on the server, the following rules apply. If the data or rule file is specified to be local (or left unspecified, in which case it is also local), skip the following and refer to [FILE-NAME](#).

If you are using server data_file or server rules_file, you can get the file from an Essbase Server catalog path, as shown in the example.

Type

name (see [MaxL Syntax Notes](#))

Example

The following statement performs a data load using a data file stored in the shared folder of the Essbase file catalog. The rule file is in the cube directory for Sample Basic.

```
import database 'Sample'. 'Basic' data from server data_file 'catalog/shared/  
Data_Basic' using server rules_file 'Data' on error write to "dataload.err";
```

For information about permitted import directories, refer to Specify Files in a Catalog Path.

Referenced By

[import data](#)

[import dimensions](#)

IMPORT-DIR

The IMPORT-DIR terminal in the MaxL language represents a directory path from which to re-import exported Essbase linked reporting objects (LROs).

 **Note:**

If importing lros from the application directory on the Essbase Server, you can specify just the directory name instead of the absolute path.

If you do not know where *<Application Directory>* is in your environment, refer to Environment Locations in the Essbase Platform.

The IMPORT-DIR string should indicate the export directory used for the [export lro](#) statement. It must be enclosed in single or double quotation marks when you pass it into the MaxL Shell.

Type

string (see [MaxL Syntax Notes](#))

Examples

Relative paths (for example, `'../exports/lros'`) are not supported.

```
'/scratch/exports/lros'
```

```
'Sample-Basic-lros'
```

For information about how IMPORT-DIR is created, see the grammar and definitions for [export lro](#).

Referenced By

[import lro](#)

JAVACLASS.METHOD

The JAVACLASS.METHOD terminal in MaxL represents the Java class and method for a custom-defined Essbase calculator function (CDF).

This string must be a fully qualified Java method name and signature, enclosed in single or double quotation marks when you pass it into the MaxL Shell.

Type

string (see [MaxL Syntax Notes](#))

Example

```
'com.hyperion.essbase.calculator.Statistics.covariance'
```

For Java code examples and MaxL registration scripts for custom-defined functions, see Custom-Defined Calculation Functions.

Referenced By

[create function](#)

LOCATION-ALIAS-NAME

The LOCATION-ALIAS-NAME terminal in the MaxL language represents the name of a location alias referencing another Essbase database.

Syntax

```
name1.name2.name3
```

- *name1*—Application name.
- *name2*—Database name.
- *name3*—Location alias name.

Type

name (see [MaxL Syntax Notes](#))

Example

```
Sample.Basic.EasternDB
```

Referenced By

[create location alias](#)

[display location alias](#)

LOC-ALIAS-SINGLE

The LOC-ALIAS-SINGLE terminal in MaxL represents the short form of the name of a location alias referencing another Essbase database.

Use this single/short form of the location alias name if you are creating a new location alias.

Type

name (see [MaxL Syntax Notes](#))

Example

```
EasternDB
```

Referenced By

[create location alias](#)

LOG-TIME

The LOG-TIME terminal in the MaxL language for Essbase represents a specific log time after which to replay subsequent transactions.

Enclose the string in quotation marks when you pass it into the MaxL Shell.

Type

string (see [MaxL Syntax Notes](#))

Example

```
'11_20_2007:12:20:00'
```

Referenced By

[alter database \(misc\)](#)

MACRO-EXPANSION

The MACRO-EXPANSION terminal in MaxL represents the extended definition of a custom-defined Essbase calculator macro (CDM), to be substituted in wherever the registered macro name is referenced in a calculation.

If the string contains special characters (see [MaxL Syntax Notes](#)), it must be enclosed in single or double quotation marks when you pass it into the MaxL Shell.

Type

string (see [MaxL Syntax Notes](#))

Example

```
'@COUNT (SKIPMISSING, @RANGE (@@S) ) '
```

See Custom-Defined Macros

Referenced By

[create macro](#)

MACRO-NAME

The MACRO-NAME terminal in MaxL represents the name of a custom-defined Essbase calculator macro (CDM). 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. Oracle recommends 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 must be enclosed in single or double quotation marks when you pass it into the MaxL Shell.

Syntax

Syntax for local (application-level) macro:

name1.name2

Syntax for global macro:

name2

- *name1*—Application name.
- *name2*—Macro name.

Type

name

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.

Referenced By

[create macro](#)

[display macro](#)

[drop macro](#)

MEMBER-EXPRESSION

The MEMBER-EXPRESSION terminal in MaxL represents a specification of members from one or more dimensions in an Essbase database outline.

The member expression can be one member, a list of members, member combinations separated by commas, or member sets defined with functions. The string must be enclosed in single or double quotation marks when you pass it into the MaxL Shell.

Type

string (see [MaxL Syntax Notes](#))

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 MEMBER-NAME terminal in MaxL represents the name of a member in an Essbase database outline.

If the member name contains special characters (see [MaxL Syntax Notes](#)), it must be enclosed in single quotation marks when you pass it into the MaxL Shell.

Type

name (see [MaxL Syntax Notes](#))

Examples

```
Jan
```

```
'New York'
```

If MEMBER-NAME is part of [MEMBER-EXPRESSION](#) and MEMBER-NAME begins with a number or contains special characters (see [MaxL Syntax Notes](#)), enclose MEMBER-NAME in double quotation marks and enclose MEMBER-EXPRESSION in single quotation marks.

To specify a member name, you can also reference:

- a member alias.
- a member ID. Example: `'id__27'`.

- an attribute member name that is differentiated using a prefix or suffix. Example:
`'Ounces_32'`.
- a qualified member name, for a member in a duplicate member name outline. Example:
`'[Ounces].[32]'`.

Referenced By[alter database](#) (set)[create partition](#)[query database](#)

OBJ-NAME

The OBJ-NAME terminal in MaxL represents the name of an Essbase database object. MaxL requires three tokens in this name: two to indicate application and database context, and one as the object name itself. For example: `appname.dbname.objectname`.

Syntax

name1.name2.name3

- *name1*—Application name.
- *name2*—Database name.
- *name3*—Object name.

Type

name (see [MaxL Syntax Notes](#))

Example

`Sample.basic.Calcdat`

Referenced By[alter object](#)[drop object](#)

OBJ-NAME-SINGLE

The OBJ-NAME-SINGLE terminal in MaxL represents the short name of an Essbase database object.

The single/short form of a stored database object name can be used when altering the object. This form of object name is equal to the third token of the full name, [OBJ-NAME](#).

If any part of the name contains special characters (see [MaxL Syntax Notes](#)), it must be enclosed in single or double quotation marks when you pass it into the MaxL Shell.

Type

name (see [MaxL Syntax Notes](#))

Example

If the full database object name is `sample.basic.calcdat`, then `OBJ-NAME-SINGLE` is `calcdat`.

Referenced By

[alter object](#)

OUTLINE-ID

The `OUTLINE-ID` terminal in MaxL represents the numeric identification of an Essbase aggregate storage (ASO) database outline associated with an aggregate 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 view selection it returns. When you materialize the views using **execute aggregate build**, you input this outline ID.

Type

number (see [MaxL Syntax Notes](#))

Example

```
4142187876
```

Referenced By

[execute aggregate selection](#)

[execute aggregate build](#)

PASSWORD

The `PASSWORD` terminal in the MaxL language represents an Essbase user's password (not applicable for externally authenticated users).

Observe these password guidelines:

- Non-Unicode application limit: 100 bytes.
- Unicode-mode application limit: 100 characters.
- If the string contains special characters (see [MaxL Syntax Notes](#)), the password must be enclosed in single or double quotation marks when you pass it into the MaxL Shell.

Use of `§` (dollar sign) character within the Essbase password is not supported for logins in a Linux environment.

- Leading or trailing spaces are illegal and will be trimmed off

Type

string (see [MaxL Syntax Notes](#))

Referenced By

[alter partition](#)

[create location alias](#)

[create outline](#)

[create partition](#)

[Login](#)

PATHNAME_FILENAME

The PATHNAME_FILENAME terminal in MaxL represents a full path to a file.

If the string contains special characters (see [MaxL Syntax Notes](#)), it must be enclosed in single or double quotation marks.

Type

string (see [MaxL Syntax Notes](#))

Referenced By

[query database](#)

PRECISION-DIGITS

The PRECISION-DIGITS terminal in MaxL represents the number of decimal places MaxL Shell should use for the Essbase data values it displays in MDX query outputs.

Precision digits must be specified as an integer between 0 and 15, inclusive.

Type

number (see [MaxL Syntax Notes](#))

Referenced By

[alter session](#)

PROPS

The PROPS terminal in MaxL represents Essbase aggregate storage (ASO) data load buffer properties that determine how missing and zero values, duplicate values, and multiple values for the same cell in the data source should be processed.

The ASO data load buffer properties, which specify how to handle empty source data values, can be specified as follows:

- `ignore_missing_values`: Ignore missing values in the source data.
- `ignore_zero_values`: Ignore zeros in the source data.
- `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](#))

Referenced By

[alter database](#) (aggregate storage)

RNUM

The RNUM terminal in the MaxL language represents a resource-usage specification for constraining temporary Essbase aggregate storage (ASO) data load buffers.

The specification 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](#))

Example

The *specified* `resource_usage` and the *actual* resource usage of the aggregate storage cache will be different, depending on how many threads you specify in `ASOCACHECONCURRENTCONSUMINGTHREADS`. The formula to calculate actual resource usage is:

$$(1/ASOCACHECONCURRENTCONSUMINGTHREADS) * resource_usage$$

The actual aggregate-storage cache size Essbase allocates, in megabytes, is calculated as:

$$ASODEFAULTCACHESIZE * actual \text{ resource usage}$$

Assume the following configurations are set for the cube:

```
ASOCACHECONCURRENTCONSUMINGTHREADS 5
ASODEFAULTTCACHESIZE 500
```

With the above settings, you can initialize up to 5 data load buffers, specifying, for each, the maximum resource_usage of 1.0.

Using the following MaxL statements, you initialize three load buffers:

```
alter database AsoSamp.Basic initialize load_buffer with buffer_id 1
resource_usage 1.0;
alter database AsoSamp.Basic initialize load_buffer with buffer_id 2
resource_usage 0.5;
alter database AsoSamp.Basic initialize load_buffer with buffer_id 3
resource_usage 0.1;
```

The following MaxL statement displays the actual resource usage:

```
query database AsoSamp.Basic list load_buffers;
buffer_id internal active resource_usage aggregation_method ignore_missings
ignore_zeros
-----+
1          FALSE    FALSE           0.2  AGGREGATE_SUM      FALSE      FALSE
2          FALSE    FALSE           0.1  AGGREGATE_SUM      FALSE      FALSE
3          FALSE    FALSE           0.02 AGGREGATE_SUM      FALSE      FALSE
```

Referenced By

[alter database](#) (aggregate storage), when initializing load buffers

RTSV-LIST

The RTSV-LIST terminal in the MaxL language represents a string of runtime substitution variables that can be used in Essbase calculation scripts for block storage (BSO) databases.

Runtime substitution variables are specified as key/value pairs. The string must be enclosed with single quotation marks, and key/value pairs must be separated by a semicolon, including a semicolon after the last runtime substitution variable in the string and before the terminal single quotation mark.

Runtime substitution variables' name and default value must be declared in the SET RUNTIMESUBVARS calculation command. If you include a runtime substitution variable in RTSV-LIST that has not been declared, Essbase ignores the undeclared runtime substitution variable (no warnings or exceptions are generated).

Type

string (see [MaxL Syntax Notes](#))

Example

In this example of a runtime substitution variable string, the name and value of four runtime substitution variables are specified (for example, the value of the runtime substitution variable named "a" is 100):

```
'a=100;b=@CHILDREN("100");c="Actual"->"Final";d="New York";'
```

Referenced By

[execute calculation](#) (block storage only)

RULE-FILE-NAME

The RULE-FILE-NAME terminal in the MaxL language represents a comma-separated list of strings of rule file names. Each rule file name in the list should be a max-8-character object file name with no extension. The rule files must reside on the Essbase Server.

Type

string (see [MaxL Syntax Notes](#))

Example

```
'h1h1h1', 'h1h1h2'
```

Referenced By

[import data](#) (aggregate storage)

SESSION-ID

The SESSION-ID terminal in MaxL represents the unique Essbase session ID. The session ID can be used to logout a user session, or end the current request in that session.

Type

number (see [MaxL Syntax Notes](#))

Example

```
3310545319
```

Referenced By

[alter system](#)

[display session](#)

[query database](#)

SIZE-STRING

The SIZE-STRING terminal in MaxL is typically a way to specify the size of an Essbase cache or memory buffer.

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](#))

Examples

```
51040b
51040 b
11MB
11000kb
12.34gb
1234e-2gb
```

Referenced By

[alter application](#)

[alter database \(set\)](#)

[alter database \(for ASO\)](#)

[alter tablespace](#)

SPOOL-NAME

The SPOOL-NAME terminal in MaxL represents the name of an Essbase database trigger's output file, located in the `trig` folder within the database directory.

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](#))

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](#)

[ACTION](#)

STOPPING-VAL

The STOPPING-VAL terminal in MaxL is a growth constraint you can specify for disk space usage when performing an aggregation on an Essbase aggregate storage (ASO) database.

This number is the 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](#))

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 TABLSP-NAME terminal in the MaxL language represents the name of a tablespace for an Essbase aggregate storage (ASO) database.

Tablespaces are applicable only to aggregate storage databases. 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](#))

Example

temp

Referenced By

[alter tablespace](#)

[display tablespace](#)

TRIGGER-NAME

The TRIGGER-NAME terminal in MaxL represents the name of the trigger device created to track and respond to Essbase 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](#))

Example

Sample.Basic.MyTrigger

Referenced By

[alter trigger](#)

[create trigger](#)

[display trigger](#)

[drop trigger](#)

URL-NAME

The URL-NAME terminal in MaxL represents the name of a drill-through URL definition used to link Essbase database values to content hosted on Oracle ERP and EPM applications.

For more about drill through URLs, see [Drill Through to a URL](#).

Syntax

```
name1.name2.name3
```

- *name1*—Application name
- *name2*—Database name
- *name3*—URL name

Type

name (see [MaxL Syntax Notes](#))

Example

```
Sample.basic.MyURL
```

If any part of the name contains special characters (see [MaxL Syntax Notes](#)), the name must be enclosed in single or double quotation marks.

Referenced By

[create drillthrough](#)

[alter drillthrough](#)

[display drillthrough](#)

[drop drillthrough](#)

USER-NAME

The USER-NAME terminal in MaxL represents the name of an Essbase user.

User name guidelines:

- Limit 50 characters
- The following special characters are not permitted:

```
; , = + * ? [ ] | < > \ " ' / # [Space] [Tab]
```

- If the user name contains any special characters (see [MaxL Syntax Notes](#)), the name must be enclosed in single or double quotation marks.

Types

- name (see [MaxL Syntax Notes](#))
- name@provider

- WITH IDENTITY [ID-STRING](#)

 **Note:**

If a user or group name includes the @ character, you must specify the provider as well. 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 \(misc\)](#)

[alter partition](#)

[alter system](#)

[create location alias](#)

[create outline](#)

[create partition](#)

[create user](#)

[display privilege](#)

[drop lock](#)

[grant](#)

[query database](#)

[Login](#)

VARIABLE-NAME

The VARIABLE-NAME terminal in MaxL represents the name of an Essbase substitution variable.

The name can only contain alphanumeric characters and the underscore: (a-z A-Z 0-9 _).

Type

name (see [MaxL Syntax Notes](#))

Example

```
curmonth
```

Referenced By

[alter application](#)

[alter database](#)

[alter system](#)

[display variable](#)

VIEW-FILE-NAME

The VIEW-FILE-NAME terminal in MaxL represents the name of an aggregation script containing information derived during aggregate view selection for an Essbase aggregate storage (ASO) database.

The file is created in the cube directory, 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](#))

Referenced By

[execute aggregate selection](#)

[execute aggregate build](#)

[query database](#)

VIEW-ID

The VIEW-ID terminal in MaxL represents the numeric identifier of an aggregate view for an Essbase aggregate storage (ASO) database.

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-IDs](#). OUTLINE-IDs change when changes are made to the outline.

Type

number (see [MaxL Syntax Notes](#))

Example

8941

Referenced By

- [execute aggregate selection](#)
- [execute aggregate build](#)

VIEW-SIZE

The VIEW-SIZE terminal in MaxL represents the approximate view size as a fraction of input data size, for an Essbase aggregate storage (ASO) database.

Express the approximate view size as a fraction of the 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](#))

Referenced By

- [execute aggregate build](#)

Privileges and Roles

Essbase privileges are grouped together into permission-sets called *roles*. Using MaxL, administrators grant roles to users depending on their access needs. The scope of a role can be the system, the application, or the database.

Essbase system privileges are indivisible database access types. In MaxL, privileges are grouped together to form permission-sets called *roles*. 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.

Table 3-19 Privileges and Roles

Privileges and Roles	read	write	calculate	manage database	create database	start application	manage application	create/drop application
no access
read	✓
write	✓	✓
execute	✓	✓	✓
manager (database)	✓	✓	✓	✓

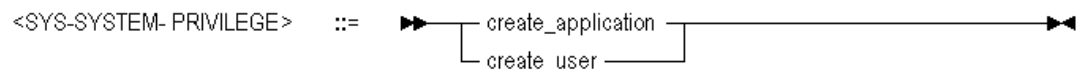
Table 3-19 (Cont.) Privileges and Roles

Privileges and Roles	read	write	calculate	manage database	create database	start application	manage application	create/drop application
manager (application)	✓	✓	✓	✓	✓	✓	✓	
administrator	✓	✓	✓	✓	✓	✓	✓	✓

System-Level System Privileges

The Essbase system level privileges in MaxL are `create_application` and `create_user`.

The following system privileges are applicable to the Essbase server. 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

The Essbase system level roles in MaxL are `no_access` and `administrator`.

The following system roles are applicable to the Essbase server. They do not apply to any specific application or database. 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.

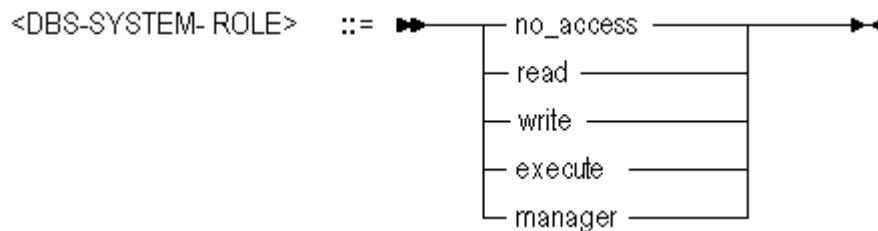
- **manager**—Manager access to the application and any databases within it. Manager access means ability to create, delete, and modify databases within the application, in addition to having Read, Write, and Execute access for that application.

Database-Level System Roles

Database-level system roles are access permissions applicable to Essbase cubes. Minimum permissions can be set at the application or cube level, affecting all users and groups. Permissions can also be granted to individual users or groups, if Essbase uses EPM Shared Services security mode.

Minimum Cube Permissions

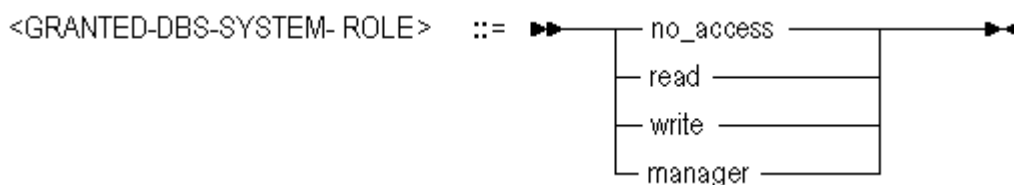
Database-level system roles are minimum access permissions you can set for Essbase cubes. The following roles have a database-wide scope and are available when assigning minimum database permissions:



- **no_access**—No access to the cube (if assigned using [alter database](#)) or to any cubes in the application (if assigned using [alter application](#)).
- **read**—Read-only access to the cube (if assigned using [alter database](#)) or to all cubes 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 cube (if assigned using [alter database](#)) or to all cubes 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 cube (if assigned using [alter database](#)) or to all cubes 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 cube (if assigned using [alter database](#)) or to all cubes in the application (if assigned using [alter application](#)). Manager access means ability to modify cube outlines, in addition to having Read and Write access.

Permissions Grantable to Users and Groups

The following database-level system roles are available for granting to users and groups, only if Essbase uses EPM Shared Services security mode.



- `no_access`—No access to the cube.
- `read`—Read-only access to the cube. Read access means ability to view files, retrieve data values, and run report scripts.
- `write`—Write access to the cube. Write access means ability to update data values, in addition to having Read access.
- `manager`—Manager access to the cube. Manager access means ability to modify cube outlines, in addition to having Read and Write access.

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](#)).

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: `display user 'O'Brien';`

Result: Error.

Example: `display user 'O\'Brien';`

Result: User `O'Brien` is displayed.

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: `display user "O'Brien";`

Result: Error.

Example: `display user "O\'Brien";`

Result: User `O'Brien` is displayed.

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:display user 'O'Brien';
```

Result: User `O'Brien` is displayed.



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

There are different ways to start (invoke) the MaxL Shell, and you may often need MaxL Shell commands in addition to MaxL statements when you work with Essbase. MaxL Shell commands include `login`, `spool`, `set column width`, `set message level`, `set timestamp`, `echo`, `nesting`, `iferror/goto`, and `logout`.

The MaxL Shell has a separate set of useful commands, independent of the MaxL language itself.

MaxL Shell Invocation

The MaxL Shell (`essmsh`) is a pre-parser mechanism for entering MaxL statements.

You can 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.

To start the `essmsh` shell, do not invoke it directly. In order for the environment to be set correctly, you must start `essmsh` using `startMAXL.bat` (Windows) or `startMAXL.sh` (UNIX).

Help is available at the operating-system command prompt if you type `startMAXL.bat -h | more`.

 **Note:**

The help text is for `essmsh` shell; however, in order for the environment to be set correctly, you must start `essmsh` using `startMAXL.bat` (Windows) or `startMAXL.sh` (UNIX). You can pass the same arguments to `startMAXL` as you would formerly pass to `essmsh`. For example, instead of `essmsh -l username password`, you should now use `startMAXL.bat -l username password`.

Interactive Input Flags

You can log into the MaxL Shell for interactive use (typing statements at the keyboard) in the following ways.

Flag	Description/Example
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 UNIX users: In the following examples, replace <code>startMAXL.bat</code> with <code>startMAXL.sh</code> . <code>startMAXL.bat</code>

Flag	Description/Example
-a Flag: Arguments	<p>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. 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).</p> <pre>startMAXL.bat -a Fiona sunflower appname dbsname</pre> <pre>MAXL> spool on to 'D:\output\createapp.out';</pre> <pre>MAXL> login \$1 identified by \$2;</pre> <pre>49 - User logged in: [Fiona].</pre> <pre>MAXL> create application \$3;</pre> <pre>30 - Application created: ['appname'].</pre> <pre>MAXL> create database \$3.\$4 as Sample.Basic;</pre> <pre>36 - Database created: ['appname'.'dbsname'].</pre> <pre>MAXL> spool off;</pre>
-l Flag: Login	<p>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.</p> <pre>startMAXL.bat -l Fiona sunflower</pre>

Flag	Description/Example
-u, -p, and -s Flags: Login Prompts and Hostname Selection	<p>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.</p> <ul style="list-style-type: none"> If -s <host-name> is passed to the shell, MaxL will prompt for the user name and password, and the password will be hidden. <pre>startMAXL.bat -s localhost Enter UserName> admin Enter Password> ***** OK/INFO - 1051034 - Logging in user admin. OK/INFO - 1051035 - Last login on Monday, January 28, 2020 10:06:16 AM. OK/INFO - 1241001 - Logged in to Essbase.</pre> 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. <pre>startMAXL.bat -u smith Enter Password > *****</pre> If -p <password> is passed to the shell and -u <username> is omitted, MaxL Shell will prompt for the user name. <pre>startMAXL.bat -p password Enter Username > smith</pre> If -m <messageLevel> is passed to the shell, only the specified level of messages will be returned by the shell. <pre>startMAXL.bat -m error</pre> <p>Values for <messageLevel> include: default, all, warning, error, and fatal. The default value is all (same as specifying default).</p>
-m Flag: Message Level	<p>If -m <messageLevel> is passed to the shell, only the specified level of messages will be returned by the shell.</p> <p>Example: <code>startMAXL.bat -m error</code></p> <p>Values for the <messageLevel> include: default, all, warning, error, and fatal. The default value is all (same as specifying default).</p>

File Input

You invoke the MaxL Shell to run scripts (instead of typing statements at the keyboard) in the following ways.

If you type `startMAXL.sh` or `startMAXL.bat` followed by a file name or path, the shell takes input from the specified file.

```
startMAXL.sh /client/scripts/filename.msh
```

Entered at the command prompt, the above example starts the shell, tells it to read MaxL statements from a file, and terminates the session when it is finished.

```
startMAXL.bat filename
```

The above example starts the shell to read MaxL statements from `filename`, located in the current directory (the directory from which the MaxL Shell was invoked).

If you type `startMAXL.sh` or `startMAXL.bat`, 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.

```
startMAXL.bat filename.msh Fiona sunflower localhost
```

The above example starts the shell to read MaxL statements from `filename.msh`, located in the current directory.

Standard Input

With the `-i` flag, `essmsh` uses standard input, which could be input from another process. For example,

```
program.sh | startMAXL.bat -i
```

When **program.sh** generates MaxL statements as output, you can pipe `program.sh` to `startMAXL.bat -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; |  
startMAXL.bat -i
```

In the above example, the MaxL Shell takes input from the `echo` command's output. User Fiona is logged in, and user privileges are displayed.

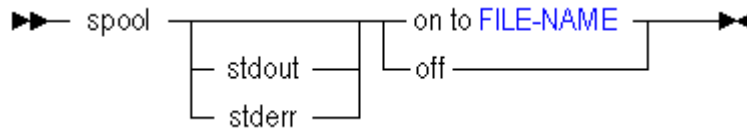
Before using any of the MaxL Shell commands that follow, you need to log in (see [Login](#))

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**.

**FILE-NAME****Example**

```
spool on to 'output.txt';
```

Sends output of MaxL statements to a file called output.txt, located in the current directory where the MaxL Shell was invoked.

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.xml 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:

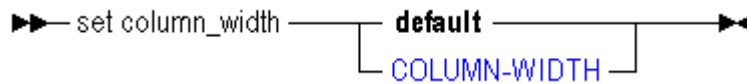
You cannot use the generic spool and the special output-channel spools in the same script. For example, the following is not valid:

```
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: None.



COLUMN-WIDTH

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.

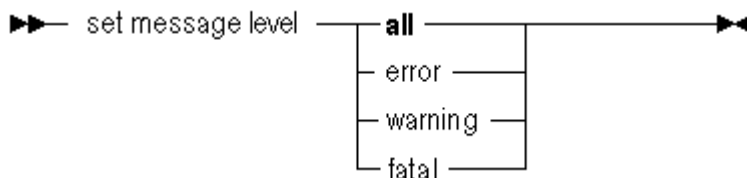


Table 3-20 MaxL Shell Message Levels

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.

```
▶▶ set timestamp on
      |
      | off
      |▶▶
```

**Note:**

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 ([MaxL Shell Syntax Rules and Variables](#)).

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;
```

Notes

Variables defined or changed in a nested script persist into the parent script after the nested script executes.

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

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 timestamp on;
iferror 'End';

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.

Version

To see which version of MaxL you are using, type **version**.

Example

```
version;
```

Logout

Log out from Essbase without exiting the interactive MaxL Shell.

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).

Example

```
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 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.

MaxL Shell Syntax Rules and Variables

The MaxL Shell requires semicolon terminators at the end of MaxL statements. You can work with variables in MaxL scripts to make them more flexible. Learn the quoting and special characters rules for MaxL Shell to ensure that your MaxL scripts for Essbase work as expected.

The MaxL Shell is a pre-parser mechanism for entering MaxL statements. The following syntax information can help you use the MaxL Shell successfully.

Semicolons

When a MaxL statement is passed to Essbase Server interactively or in batch mode via the MaxL Shell, 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, do *not* use semicolons.

Table 3-21 Semicolon Usage Examples in MaxL

Program	Example
Interactive MaxL Shell	<code>create application Sample;</code>
MaxL Shell script:	<pre>login \$1 identified by \$2; create application Sample; create currency database Sample.Interntl; display database Sample.Interntl; exit;</pre>

Variables

Overview of Variables in MaxL Shell

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`" could be interpreted as `/scratch/user/oracle_home/essbase/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 `/scratch/user/oracle_home/essbase/products/Essbase/EssbaseServer`.

Environment Variables

You can reference any environment variable in the MaxL Shell.

Example (Unix): `spool on to "$ESSBASEPATH\out.txt"`;

Result: MaxL Shell session is recorded to `/scratch/user/oracle_home/essbase/products/Essbase/EssbaseServer/out.txt` .

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" */
```

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.

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*.

```
essmsh -a Fiona sunflower sample basic login $1 $2;
```

Table 3-22 Quotation Marks' Usage and Effect on Variables in MaxL

Example	Return Value	Explanation
<code>echo \$1;</code>	Fiona	\$1 is expanded as the first invocation argument.
<code>echo "\$1's hat";</code>	Fiona's hat	\$1 is expanded as the first invocation argument, and the special character ' is allowed because double quotation marks are used.
<code>echo \$3;</code>	sample	\$3 is expanded as the third invocation argument.
<code>echo '\$3';</code>	\$3	\$3 is taken literally and not expanded, because it is protected by single quotation marks.
<code>display database \$3.\$4;</code>	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 .
<code>echo "\$3.\$4";</code>	sample.basic, but interpreted as one token (NOT suitable for DBS-NAME , 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

These rules are for MaxL Shell commands. Applicable commands include `spool on/off`, `echo`, and `nesting`.

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 `/scratch/user/oracle_home/essbase/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 "$APPDIR\app\sample-basic-lros";`

Result: Error. Import is a MaxL statement, and for MaxL, `'\'` is ignored.

Example: `import database sample.basic lro from directory "$APPDIR\\app\\sample-basic-lros";`

Result: MaxL imports LRO information to `Sample.Basic` from `$APPDIR\app\sample-basic-lros` (if you have an `APPDIR` variable defined).

Query Cancellation

You may need to cancel a MaxL query to Essbase.

To cancel a query running from MaxL Shell, use the Esc key.

Encryption

You can encrypt Essbase user and password information stored in MaxL scripts, using public and private keys.

The following MaxL Shell invocation generates a public-private key pair that you can use to encrypt a MaxL script.

Linux

```
startMAXL.sh -gk
```

Windows

```
startMAXL.bat -gk
```

The following MaxL Shell invocation encrypts the input MaxL script, obscuring user name and password, and changing the file extension to .mxls.

Linux

```
startMAXL.sh -E scriptname.xml PUBLIC-KEY
```

Windows

```
startMAXL.bat -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.

Linux

```
startMAXL.sh -D scriptname.mxls PRIVATE-KEY
```

Windows

```
startMAXL.bat -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.

Linux

```
startMAXL.sh -ep DATA PUBLIC-KEY
```

Windows

```
startMAXL.bat -ep DATA PUBLIC-KEY
```

The following invocation enables you to encrypt the base script while saving any nested scripts for manual encryption.

Linux

```
startMAXL.sh -Em scriptname.xml PUBLIC-KEY
```

Windows

```
startMAXL.bat -Em scriptname.xml PUBLIC-KEY
```

LoginAs

To facilitate creating scheduled reports with user-appropriate permissions, Essbase administrators can use LoginAs to impersonate another user during MaxL login.

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
```

Login

Before you can send MaxL statements from the MaxL Shell to Essbase, you must log in to an Essbase Server session.

As a prerequisite to using MaxL, follow the client setup instructions in [Manage Essbase Using the MaxL Client](#).



Note:

Before logging in to an Essbase Server session, you must start the MaxL Shell (see MaxL Invocation Summary, in [MaxL Shell Commands](#)). Or, you can start the MaxL Shell and log in at the same time (see `-l Flag: Login` at the same link).

```
login USER-NAME identified by PASSWORD on HOST-NAME
```

- [USER-NAME](#)
- [PASSWORD](#)
- [HOST-NAME](#)

Example

For an independent deployment:

```
login admin pa5sw0rd on "https://myserver.example.com:9001/essbase/agent";
```

For a stack deployment on OCI:

```
login admin pa5sw0rd on "https://192.0.2.1:443/essbase/agent";
```



Note:

The Essbase system administrator logging in must have Identity Domain Administrator and Security Administrator role in the confidential identity application.

ESSCMD Script Conversion

Use the Essbase `cmd2mxl` utility if you need to convert ESSCMD shell scripts to MaxL scripts.

`cmd2mxl` is a fully supported utility for converting existing ESSCMD shell scripts to their corresponding MaxL scripts. To convert an ESSCMD shell script to a MaxL script, go to the operating-system command prompt and enter the executable name, the ESSCMD shell script name, the desired MaxL script name, and the name of a logfile to write to in case of errors.

- [ESSCMD Script Utility Usage](#)
- [Things to Note About the ESSCMD shell Script Utility](#)
- [ESSCMD to MaxL Mapping](#)

ESSCMD Script Utility Usage

```
cmd2mxl esscmd_script maxl_output logfile
```

For example, if the ESSCMD shell 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 shell Script Utility

1. The utility will only translate syntactically and semantically valid ESSCMD shell scripts.
2. For invalid ESSCMD shell scripts, the resulting MaxL script is undefined.
3. All ESSCMD shell statements in the scripts should end with a semicolon (;) statement terminator.
4. This utility will only work on Windows platforms.
5. Although most ESSCMD shell 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 shell command is still needed, it can be invoked from a MaxL script using `shell esscmd <scriptname>`.
6. All strings in the ESSCMD shell scripts should be surrounded by double quotation marks (").

ESSCMD to MaxL Mapping

The following table compares ESSCMD shell usage to MaxL usage, and the following conversions are supported by **cmd2mxl**.

Table 3-23 ESSCMD shell to MaxL Mapping

ESSCMD shell Command	ESSCMD shell Usage Example	MaxL Equivalent Example
ADDUSER	ADDUSER finance essexer1;	N/A. User management statements no longer supported in MaxL.
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.rul' on error write to 'c:\error.log';
BUILDDIM	builddim 1 "c:\data_rul.rul" 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;

Table 3-23 (Cont.) ESSCMD shell to MaxL Mapping

ESSCMD shell Command	ESSCMD shell Usage Example	MaxL Equivalent Example
CREATEGROUP	creategroup managers;	N/A. User management statements no longer supported in MaxL.
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;	N/A. User management statements no longer supported in MaxL.
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;	N/A. User management statements no longer supported in MaxL.
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;	N/A. User management statements no longer supported in MaxL.
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;

Table 3-23 (Cont.) ESSCMD shell to MaxL Mapping

ESSCMD shell Command	ESSCMD shell 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;
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

Table 3-23 (Cont.) ESSCMD shell to MaxL Mapping

ESSCMD shell Command	ESSCMD shell Usage Example	MaxL Equivalent Example
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 lro 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;
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 shell 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';

Table 3-23 (Cont.) ESSCMD shell to MaxL Mapping

ESSCMD shell Command	ESSCMD shell Usage Example	MaxL Equivalent Example
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;	N/A. User management statements no longer supported in MaxL.
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;	N/A. User management statements no longer supported in MaxL.
RESETDB	select sample basic; resetdb;	alter database sample.basic reset;
RESETPERFSTATS	resetperfstats enable;	alter database sample.basic set performance statistics enabled;
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';

Table 3-23 (Cont.) ESSCMD shell to MaxL Mapping

ESSCMD shell Command	ESSCMD shell Usage Example	MaxL Equivalent Example
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';
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.

Table 3-23 (Cont.) ESSCMD shell to MaxL Mapping

ESSCMD shell Command	ESSCMD shell Usage Example	MaxL Equivalent Example
SETMSGLEVEL	setmsglevel 2;	set message level all;
SETPASSWORD	setpassword steve newpass;	N/A. User management statements no longer supported in MaxL.
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';

 **Note:**

This is part of the separate MaxL Shell grammar, not the MaxL language itself.

MaxL Reserved Words List

The following keywords are part of the MaxL grammar, and are reserved. If you intend to use any of these words as names or passwords in Essbase, 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
catalog
cell_status
change_file
clear
client
cnt_sempaphore
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
sourcereion
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

MaxL BNF

MaxL BNF notation is an optional alternative to railroad diagrams, for learning MaxL syntax for Essbase.

Key

```
{ }      Alternatives (at least one required)
[ ]      Options (none required)
!!       Default option if none indicated
|        Separates options (OR)
[,...]  Comma-separated list (of previous item) allowed
[ ...]  Whitespace-separated list (of previous item) allowed
''       Literal
::=     "is defined as." Symbol to the left is to be replaced with expression
on the right
TERMINAL
%NON-TERMINAL%
```

alter application

```
alter application
{APP-Name
  {set
    {lock_timeout after INTEGER[!seconds!|minutes]
      |max_lro_file_size {unlimited|SIZE-STRING}
      |minimum permission %DBS-SYSTEM-ROLE%
      |variable VARIABLE-NAME STRING
      |cache_size SIZE-STRING
      |type unicode_mode
    }
    |{load|unload} database DBS-STRING
    |{enable|disable} {startup|autostartup|commands|updates|connects|security}
    |comment COMMENT-STRING
    |clear logfile
    |add variable VARIABLE-NAME [STRING]
    |drop variable VARIABLE-NAME
    |rename to APP-NAME
  }
}

DBS-SYSTEM-ROLE::=
{no_access|read|write|execute|manager}
```

alter application (aggregate storage)

```
alter application
{APP-Name
  {set
    {
      |minimum permission %DBS-SYSTEM-ROLE%
      |variable VARIABLE-NAME STRING
      |cache_size SIZE-STRING
    }
  }
}
```

```

    |type unicode_mode
  }
|{load|unload} database DBS-STRING
|{enable|disable} {startup|autostartup|commands|updates|connects|security}
|comment COMMENT-STRING
|clear logfile
|add variable VARIABLE-NAME [STRING]
|drop variable VARIABLE-NAME
|rename to APP-NAME
}

DBS-SYSTEM-ROLE ::=
{no_access|read|write|execute|manager}

```

alter database enable|disable

```

alter database DBS-NAME
{enable|disable}
{
  two_pass_calc
  |aggregate_missing
  |startup
  |autostartup
  |compression
  |create_blocks
  |committed_mode
  |pre_image_access
}

```

alter database set

```

alter database DBS-NAME
set
{
  retrieve_buffer_size SIZE-STRING
  |retrieve_sort_buffer_size SIZE-STRING
  |data_cache_size SIZE-STRING
  |index_cache_size SIZE-STRING
  |currency_database DBS-STRING
  |currency_member MEMBER-NAME
  |currency_conversion {division|multiplication}
  |minimum_permission %DBS-SYSTEM-ROLE%
  |compression {rle|bitmap}
  |lock_timeout
  {
    immediate
    |never
    |after INTEGER {[!seconds!|minutes]}
  }
  |implicit_commit after INTEGER {blocks|rows}
  |variable VARIABLE-NAME STRING
  |default_calculation {CALC-NAME-SINGLE|as calc_string CALC-STRING}
  |active_alias_table ALT-NAME-SINGLE
  |performance_statistics {enabled|disabled|mode to %PST-SPEC%}
  |note COMMENT-STRING
}

```



```

    }

DBS-SYSTEM-ROLE::=
  {no_access|read|write|execute|manager}

PST-SPEC::=
  {
    default
    |{medium|long} persistence {all|database|server} scope
  }

```

alter database misc

```

alter database DBS-NAME
{
  reset [{all|data}]
  |validate
  {
    data to local logfile FILE-NAME
    |using {error_file FILE-NAME|default error_file}
  }
  |force restructure
  |load alias_table ALT-NAME-SINGLE from data_file FILE-NAME
  |unload alias_table ALT-NAME-SINGLE
  |add variable VARIABLE-NAME [STRING]
  |drop variable VARIABLE-NAME
  |delete lro
  {
    all
    |by USER-NAME
    |before DATE
    |by USER-NAME before DATE
  }
  |unlock all objects
  |begin archive to file FILE-NAME
  |end archive
  |[force] archive to file FILE-NAME
  |[force] restore from file FILE-NAME
  |replay transactions
  {
    after LOG-TIME
    |using sequence_id_range ID-RANGE
  }
  |rename to DBS-STRING
  |comment COMMENT-STRING
}

```

alter database disk volumes

```

alter database DBS-NAME
{
  {add|drop} disk volume VOLUME-NAME
  |set disk volume VOLUME-NAME
  {
    file_type {data|index|index_data}
  }
}

```

```

    |file_size SIZE-STRING
    |partition_size {SIZE-STRING|unlimited}
  }
}

```

alter database (aggregate storage)

```

alter database DBS-NAME
{
  {enable|disable}
  {
    startup
    |autostartup
    |query_tracking
    |replication_assume_identical_outline
  }
  |set
  {
    retrieve_buffer_size SIZE-STRING
    |retrieve_sort_buffer_size SIZE-STRING
    |minimum_permission %DBS-SYSTEM-ROLE%
    |variable VARIABLE-NAME STRING
    |active_alias_table ALT-NAME-SINGLE
  }
  |reset [{all|data}]
  |clear
  {
    aggregates
    |data in region CUBE-AREA[,...][physical]
  }
  |compact outline
  |add variable VARIABLE-NAME [STRING]
  |drop variable VARIABLE-NAME
  |%LOAD-BUFFER-INIT%
  |destroy load_buffer with buffer_id BUFFER-ID[,...]
  |unlock all objects
  |rename to DBS-STRING
  |comment COMMENT-STRING
  |merge {all|incremental} data
  |begin archive to file FILE-NAME
  |end archive
}

DBS-SYSTEM-ROLE::=
{no_access|read|write|execute|manager}

LOAD-BUFFER-INIT::=
initialize load_buffer with buffer_id BUFFER-ID[,...]
  [resource_usage RNUM] [property PROPS] [wait_for_resources]

```

alter drillthrough

```

alter drillthrough
URL-NAME from xml_file FILE-NAME

```

```
on {'MEMBER-EXPRESSION'}[,...]  
[allow_merge]
```

alter filter

```
alter filter FILTER-NAME  
add {no_access|read|write|meta_read} on MEMBER-EXPRESSION [,...]
```

alter group

```
alter group GROUP-NAME revoke filter FILTER-NAME
```

alter object

```
alter object OBJ-NAME of type %OBJ-TYPE%  
{rename to OBJ-NAME-SINGLE|unlock|[force]copy to OBJ-NAME}
```

```
OBJ-TYPE::=  
outline  
|calc_script  
|report_file  
|rules_file  
|text  
|partition_file  
|lro  
|selection  
|wizard  
|eqd  
|outline_paging_file  
|worksheet  
|alias_table
```

alter partition

```
alter {transparent|replicated} partition DBS-NAME  
{to|from} DBS-NAME [at HOST-NAME]  
set{  
  connect as USER-NAME identified by PASSWORD  
  |hostname as HOST-NAME instead of HOST-NAME direction {single|all}  
  |application as APP-NAME instead of APP-NAME direction {single|all}  
  |database as DSB-STRING instead of DSB-STRING  
}
```

alter session

```
alter session set dml_output  
{  
  [  
    !default!  
    |alias {on|off}  
    |metadata_only {on|off}  
    |cell_status {on|off}  
    |numerical_display {!default!|fixed_decimal|scientific_notation}
```

```

|precision PRECISION-DIGITS]
|formatted_value {on|off}
|get_missing_cells {on|off}
|get_meaningless_cells {on|off}
[,...]
}

```

alter system

```

alter system
{
load application {all|APP-NAME}
|unload application {all|APP-NAME} [no_force]
|set
{
session_idle_limit {INTEGER[!seconds!|minutes]|none}
|session_idle_poll {INTEGER[!seconds!|minutes]|none}
|invalid_login_limit {INTEGER|none|}
|inactive_user_days {INTEGER[days]|none}
|password_reset_days {INTEGER[days]|none}
|variable VARIABLE-NAME STRING
|server_port begin at INTEGER end at INTEGER
}
|delete export_directory EXPORT-DIR
|add variable VARIABLE-NAME[STRING]
|drop variable VARIABLE-NAME
|logout session %SESSION-SPEC% [force]
|shutdown
|kill request %SESSION-SPEC%
|{enable|disable} unicode
|reconcile[force]
}

```

```

SESSION SPEC ::=
all
|SESSION-ID
|by user USER-NAME
[
on application APP-NAME
|on database DBS-NAME
]
|on application APP-NAME
|on database DBS-NAME

```

alter system (aggregate storage)

```

alter system
{
load application {all|APP-NAME}
|unload application {all|APP-NAME} [no_force]
|set

```

```

    {
      session_idle_limit {INTEGER[!seconds!|minutes]|none}
      |session_idle_poll {INTEGER[!seconds!|minutes]|none}
      |invalid_login_limit {INTEGER|none}
      |inactive_user_days {INTEGER[days]|none}
      |password_reset_days {INTEGER[days]|none}
      |variable VARIABLE-NAME STRING
      |server_port begin at INTEGER end at INTEGER
    }
  |add variable VARIABLE-NAME[STRING]
  |drop variable VARIABLE-NAME
  |logout session %SESSION-SPEC% [force]
  |shutdown
  |kill request %SESSION-SPEC%
  |reconcile [force]
}

```

SESSION SPEC::=

```

all
|SESSION-ID
|by user USER-NAME
[
  on application APP-NAME
  |on database DBS-NAME
]
|on application APP-NAME
|on database DBS-NAME

```

alter tablespace (aggregate storage)

```

alter tablespace TABLSP-NAME
{
  add file_location FILE-NAME
  [
    set max_file_size SIZE-STRING
    |set max_disk_size SIZE-STRING
    [,...]
  ]
  |alter file_location FILE-NAME
  [
    set max_file_size SIZE-STRING
    |set max_disk_size SIZE-STRING
    [,...]
  ]
  |drop file_location FILE-NAME
}

```

alter trigger

```

alter trigger
{
  TRIGGER-NAME {enable|disable}
}

```

```
    |on database DBS-NAME disable  
  }
```

alter user

```
alter user  
{  
  USER-NAME  
  {  
    |revoke filter FILTER-NAME  
  }  
}
```

create application

```
create [or replace] application APP-NAME  
  [type {!nonunicode_mode!|unicode_mode}]  
  [as APP-NAME]  
  [comment COMMENT-STRING]
```

create application (aggregate storage)

```
create [or replace] application APP-NAME  
  [type {!nonunicode_mode!|unicode_mode}]  
  [using aggregate_storage]  
  [as APP-NAME]  
  [comment COMMENT-STRING]
```

create calculation

```
create [or replace] calculation CALC-NAME {CALC-STRING|as CALC-NAME}
```

create database

```
create [or replace] [currency] database DBS-NAME  
  [using non_unique_members]  
  [as DBS-NAME]  
  [comment COMMENT-STRING]
```

create database (aggregate storage)

```
create [or replace] database DBS-NAME  
  [using non_unique_members]  
  [comment COMMENT-STRING]
```

create drillthrough

```
create drillthrough URL-NAME from xml_file FILE-NAME  
  on {'MEMBER-EXPRESSION [,...]}'  
  [level0 only]
```

create filter

```
create [or replace] filter FILTER-NAME
{
  as FILTER-NAME
  |
  {
    no_access
    |read
    |write
    |meta_read
  }
  on MEMBER-EXPRESSION
  [, ...]
}
[definition_only]
```

create function

```
create [or replace] function FUNC-NAME
as JAVACLASS.METHOD
[spec CALC-SPEC-STRING
 [comment COMMENT-STRING]
]
[with property runtime]
```

create group

```
create group GROUP-NAME
 [type external]
```

create location alias

```
create [or replace] location alias
{
  LOC-ALIAS-SINGLE from DBS-NAME
  |LOCATION-ALIAS-NAME
}
to DBS-NAME at HOST-NAME as USER-NAME identified by PASSWORD
```

create macro

```
create [or replace] macro MACRO-NAME
 [%MACRO-SIGNATURE%]
as MACRO-EXPANSION
 [spec CALC-SPEC-STRING [comment COMMENT-STRING]]
```

```
MACRO-SIGNATURE ::=
 '('
 {
  !any!
  |single

```

```

    |group
    |optional
    |optional_group
    [,...]
  }
  ')'

```

create replicated partition

```

create [or replace] replicated partition DBS-NAME
%AREA-SPEC%
{to|from}
DBS-NAME [at HOST-NAME][as USER-NAME identified by PASSWORD]
[using USER-NAME identified by PASSWORD for creation]
[%AREA-SPEC%]
[
  mapped
  {globally|AREA-ALIAS}
  '('MEMBER-NAME [,...]'
  to '('MEMBER-NAME [,...]'
  [,...]
]
[outline {!direct!|reverse}]
[comment COMMENT-STRING]
[remote comment COMMENT-STRING]
[update {allow|disallow}]
[validate only]

AREA-SPEC::=
area MEMBER-EXPRESSION [AREA-ALIAS] [ ...]

```

create transparent partition

```

create [or replace] transparent partition DBS-NAME
%AREA-SPEC%
{to|from}
DBS-NAME [at HOST-NAME][as USER-NAME identified by PASSWORD]
[using USER-NAME identified by PASSWORD for creation]
[%AREA-SPEC%]
[
  mapped
  {globally|AREA-ALIAS}
  '('MEMBER-NAME [,...]'
  to '('MEMBER-NAME [,...]'
  [,...]
]
[outline {!direct!|reverse}]
[comment COMMENT-STRING]
[remote comment COMMENT-STRING]
[validate only]

AREA-SPEC::=
area MEMBER-EXPRESSION [AREA-ALIAS] [ ...]

```


create after-update trigger

```
create [or replace] after update trigger TRIGGER-NAME
  where CUBE-AREA [when CONDITION then ACTION][ ...] end
```

create on-update trigger

```
create [or replace] [!on update!] trigger TRIGGER-NAME
  [log_value {!OFF!|ON}]
  where CUBE-AREA
  [when CONDITION then ACTION][ ...]
  [else ACTION]
  end
```

create user

```
create user USER-NAME %EXTERNAL-USER-SPEC%
EXTERNAL-USER-SPEC:=
  type external
```

display application

```
display application [!all!|APP-NAME [message_level]]
```

display calculation

```
display calculation
[
  !all!
  |CALC-NAME
  |on application APP-NAME
  |on database DBS-NAME
]
```

display database

```
display database
[
  !all!
  |DBS-NAME
  |on application APP-NAME
]
[request_history]
```

display disk volume

```
display disk volume
[!all!|UNIQUE-VOL-NAME|on database DBS-NAME]
```

display drillthrough

```
display drillthrough
{
  DBS-NAME [to FILE-NAME-PREFIX]
  |URL-NAME [to FILE-NAME]
}
```

display filter

```
display filter [!all!|FILTER-NAME|on database DBS-NAME]
```

display filter row

```
display filter row [!all!|FILTER-NAME|on database DBS-NAME]
```

display function

```
display function [!all!|on system|on application APP-NAME|FUNC-NAME]
```

display location alias

```
display location alias [!all!|LOCATION-ALIAS-NAME|on application APP-NAME|on
database DBS-NAME]
```

display lock

```
display lock [!all!|on system|on application APP-NAME|on database DBS-NAME]
```

display macro

```
display macro [!all!|on system|on application APP-NAME|MACRO-NAME]
```

display object

```
display [locked] object
[
  [!all!|of type %OBJ-TYPE%]
  [!on system!|on application APP-NAME|on database DBS-NAME]
  |OBJ-NAME of type %OBJ-TYPE%
]
```

```
OBJ-TYPE::=
outline
|calc_script
|report_file
|rules_file
|text
|partition_file
|lro
```

```
|selection  
|wizard  
|eqd  
|outline_paging_file  
|worksheet  
|alias_table
```

display partition

```
display partition [!all!|on database DBS-NAME][advanced]
```

display privilege

```
display privilege  
{  
  user [!all!|USER-NAME]  
  |group [!all!|GROUP-NAME]  
}
```

display session

```
display session  
[  
  !all!  
  |SESSION-ID  
  |by user USER-NAME [on application APP-NAME|on database DBS-NAME]  
  |on application APP-NAME  
  |on database DBS-NAME  
]
```

display system

```
display system  
[  
  version  
  |ports {in use|overview}  
  |export_directory  
  |license_info  
  |security mode  
  |configuration  
  {  
    |agent  
    |network  
    |errors  
    |on database DBS-NAME  
  }  
  |message_level  
]
```

display trigger

```
display trigger  
[
```

```
!all!  
|on system  
|on application APP-NAME  
|on database DBS-NAME  
|TRIGGER-NAME  
]
```

display trigger spool

```
display trigger_spool  
[  
!all!  
|on application APP-NAME  
|on database DBS-NAME  
|SPOOL-NAME  
]
```

display variable

```
display variable  
[  
!all!  
|VARIABLE-NAME  
|on application APP-NAME  
|on database DBS-NAME  
|on system  
]
```

drop application

```
drop application APP-NAME [cascade] [force]
```

drop calculation

```
drop calculation CALC-NAME
```

drop database

```
drop database DBS-NAME [force]
```

drop drillthrough

```
drop drillthrough URL-NAME
```

drop filter

```
drop filter FILTER-NAME
```

drop function

```
drop function FUNC-NAME
```

drop group

```
drop group GROUP-NAME
```

drop location alias

```
drop location alias LOCATION-ALIAS-NAME
```

drop lock

```
drop lock
[
  !all!
  |[
    !on system!
    |on application APP-NAME
    |on database DBS-NAME
  ]
  [!all!|held by USER-NAME]
]
```

drop macro

```
drop macro MACRO-NAME
```

drop object

```
drop object OBJ-NAME of type %OBJ-TYPE% [force]
```

```
OBJ-TYPE::=
outline
|calc_script
|report_file
|rules_file
|text
|partition_file
|lro
|selection
|wizard
|eqd
|outline_paging_file
|worksheet
|alias_table
```

drop partition

```
drop
  {transparent|replicated}
  partition DBS-NAME {from|to} DBS-NAME
  [at HOST-NAME][force]
```

drop trigger

```
drop trigger TRIGGER-NAME
```

drop trigger spool

```
drop trigger_spool {SPOOL-NAME|all on database DBS-NAME}
```

drop user

```
drop user USER-NAME
```

execute aggregate build

```
execute aggregate build on database DBS-NAME
using
  {
    views VIEW-ID VIEW-SIZE [,...] with outline_id OUTLINE-ID
    |view_file VIEW-FILE-NAME
  }
```

execute aggregate process

```
execute aggregate process on database DBS-NAME
  [stopping when total_size exceeds STOPPING-VAL]
  [based on query_data]
  [{enable|!disable!} alternate_rollups]
```

execute aggregate selection

```
execute aggregate selection on database DBS-NAME
  [
    using views VIEW-ID[,...]
    with outline_id OUTLINE-ID
    [[!suppress|!force] display]
  ]
  [selecting INTEGER views]
  [stopping when total_size exceeds STOPPING-VAL]
  [based on query_data]
  [{dump|force_dump} to view_file VIEW-FILE-NAME]
  [{enable|!disable!} alternate_rollups]
```

execute allocation (aggregate storage)

```

execute allocation process on database DBS-NAME with
{
  pov MDX-SET
  amount ALLOC-NUMERIC
  {
    [amountcontext MDX-TUPLE]
    [amounttimespan MDX-SET ]
  }
  target MDX-TUPLE
  {
    [targettimespan MDX-SET]
    [targettimespanoptions {!divideamount!|repeatamount}]
    [offset MDX-TUPLE]
    [debitmember MDX-MBR]
    [creditmember MDX-MBR]
  }
  range MDX-SET
  {
    [excludedrange MDX-SET]
    [basis MDX-TUPLE]
    [basistimespan MDX-SET]
    [basistimespanoptions {splitbasis|combinebasis}]
    [
      share
      |spread [{skip_missing|skip_zero|skip_negative},...]
    ]
    [zeroamountoptions {skip_to_next_amount|abort}]
    [zerobasisoptions
      {
        skip_to_next_amount
        |abort
      }
    ]
    [negativebasisoptions
      {
        skip_to_next_amount
        |abort
        |absolute_value
        |missing_value
        |zero_value
      }
    ]
  }
  [round
    {INTEGER|MDX-NUMERIC}
    {
      discard errors
      |errors_to_lowest
      |errors_to_highest
      |errors_to_location MDX-TUPLE
    }
  ]
}

```

```
    [{!override!|add|subtract} values]
  }
```

execute calculation

```
execute calculation
{
  CALC-NAME
  |CALC-NAME on database DBS-STRING
  |{CALC-STRING|default} on DBS-NAME
}
```

execute calculation (aggregate storage)

```
execute calculation on database DBS-NAME with
local script_file FILE-NAME pov MDX-SET sourceregion MDX-SET
{
  [target MDX-TUPLE]
  |[debitmember MDX-MBR]
  |[creditmember MDX-MBR]
  |[offset MDX-TUPLE]
}
[!override!|add|subtract} values]
```

export data

```
export database DBS-NAME
{
  [!all!|level0|input]
  data [anonymous] [in columns] to [!server!] data_file FILE-NAME[,...]
  |using [!local!|server] report_file FILE-NAME to data_file FILE-NAME
}
```

export data (aggregate storage)

```
export database DBS-NAME
{
  [!level0!|input]
  data [anonymous] to [!server!] data_file FILE-NAME[,...]
  |using [!local!|server] report_file FILE-NAME to data_file FILE-NAME
}
```

export lro

```
export databse DBS-NAME lro to
[!server!|local] directory
{DBS-EXPORT-DIR|FULL-EXPORT-DIR}
```

export outline

```
export outline {DBS-NAME|FILE-NAME}
{
```



```

    all dimensions
    |list dimensions {'DIM-NAME'}[,...]
  }
  [tree|with alias_table ALT-NAME-SINGLE]
  to xml_file FILE-NAME

```

export query_tracking

```

export query_tracking DBS-NAME to
  [!server!] file FILE-NAME

```

grant

```

grant
{
  {create_application|create_user|no_access|administrator}
  [on system]
  |{no_access|manager} on application APP-NAME
  |{no_access||read|write|manager} on database DBS-NAME
  |filter FILTER-NAME
  |execute
  {
    CALC-NAME
    |{any|default}
    [
      !on system!
      |on application APP-NAME
      |on database DBS-NAME
    ]
  }
  to {USER-NAME|GROUP-NAME}

```

import data

```

import database DBS-NAME
  [using max_threads INTEGER]
  data
  {
    from
      [!local!|server]
      [!text!]
      data_file IMP-FILE
      [using [!local!|server] rules_file IMP-FILE]
      |from data_string STRING
      |connect as SQL-USR identified by SQL-PASS
      using [!local!|server] rules_file IMP-FILE
  }
  on error {{write|append}to FILE-NAME|abort}

```

import data (aggregate storage)

```

import database DBS-NAME data
{

```

```

from
  [!local!|server]
  [!text!]
  data_file IMP-FILE
  [using [!local!|server] rules_file IMP-FILE]
|from data_string STRING
|connect as SQL-USR identified by SQL-PASS
using
{
  [!local!|server] rules_file IMP-FILE
  |multiple rules_file RULE-FILE-NAME[,...]
  to load_buffer_block starting with buffer id BUFFER-ID
  on error {write to FILE-NAME|abort}
}
|from load_buffer with buffer_id BUFFER-ID[,...]
[!{override!|add|subtract} values]
[create slice]
|override {all|incremental} data

]
}
on error {{write|append}to FILE-NAME|abort}

```

import dimensions

```

import database DBS-NAME dimensions
{
  from
    [!local!|server]
    [!text!] data_file IMP-FILE
    using[!local!|server] rules_file IMP-FILE
    [!{enforce!|suppress} verification]
    |connect as SQL-USR identified by SQL-PASS
    using[!local!|server] rules_file IMP-FILE
  }[,...]
  [
    !preserve all data!
    |preserve {level0|input} data
  ]
  on error {write|append} to FILE-NAME
}

```

import lro

```

import database DBS-NAME
lro from [!local!|server] directory IMPORT-DIR

```

import query_tracking

```

import query_tracking DBS-NAME from
  [!server!] file FILE-NAME

```

query application

```
query application APP-NAME get cache_size
```

query application (aggregate storage)

```
query application APP-NAME
{
  get cache_size
  |list aggregate_storage storage_info
}
```

query archive file

```
query archive_file FILE-NAME {get overview|list disk volume}
```

query database

```
query database DBS-NAME
{
  get
  {
    active_alias_table
    |attribute_info MEMBER-NAME
    |attribute_spec
    |currency_rate
    |dbstats {dimension|data_block}
    |default calculation
    |member_info MEMBER-NAME
    |member_calculation MEMBER-NAME
    |estimated size
    |performance statistics
    {
      kernel_io
      |kernel_cache
      |end_transaction
      |database_synch
      |database_asynch
      |dynamic_calc
    }
  }
  table
}
|list
{
  alias_table
  |alias_names in alias_table ALT-NAME-SINGLE
  |lro
  [
    !all!
    |by USER-NAME
    |before DATE
    |by USER-NAME before DATE
  ]
}
```

```

    |{all|data|index} file information
    |transactions
    [
        after LOG-TIME
        [[force] write to file PATHNAME_FILENAME]
    ]
}
}

```

query database (aggregate storage)

```

query database DBS-NAME
{
    get
    {
        active_alias_table
        |attribute_info MEMBER-NAME
        |attribute_spec
        |cube_size_info
        |dbstats {dimension|data_block}
        |member_info MEMBER-NAME
        |opg_state of %OPG-SECTION% for dimension DIM-NAME
    }
    |list
    {
        |aggregate_storage runtime_info
        |aggregate_storage compression_info
        |aggregate_storage group_id_info
        |aggregate_storage slice_info
        |aggregate_storage uncommitted_transaction_info
        |alias_table
        |alias_names in alias_table ALT-NAME-SINGLE
        |existing_views [based on query_data
        |{!all!|data|index} file information
        |load_buffers
        |aso_level_info
    }
    |{dump|force_dump}
        existing_views to view_file VIEW-FILE-NAME
        [based on query_data]
    }
}

```

refresh custom definitions

```

refresh custom definitions on application APP-NAME

```

refresh outline

```

refresh outline on {transparent|replicated}
partition DBS-NAME {to|from} DBS-NAME
[at HOST-NAME]
{
    purge outline change_file
    |apply all
}

```

```

    |apply nothing
    |%OTL-CHANGE-SPEC%
}

OTL-CHANGE-SPEC::=
apply on dimension
  {add|delete|rename|update|move}[... ,]
apply on member
  {add|delete|rename|move}[... ,]
apply on member_property {
  account_type
  |alias
  |calc_formula
  |consolidation
  |currency_conversion
  |currency_category
  |data_storage
  |uda
}[... ,]

```

refresh replicated partition

```

refresh replicated partition DBS-NAME
{to|from} DBS-NAME
[at HOST-NAME]
[![all!|updated]data]

```

MaxL Use Cases

The following use cases demonstrate some Essbase tasks that can be streamlined using MaxL: creating an aggregate storage database, loading data into aggregate storage database using buffers, deleting a dangling partition definition, designing security filters based on metadata, and working with triggers to track state changes to a cube area.

- [Creating an Aggregate Storage Sample Using MaxL](#)
- [Load Data Using Buffers](#)
- [List Aggregate Storage Data Load Buffers](#)
- [Force Deletion of Partitions](#)
- [Metadata Filtering](#)
- [Examples of Triggers](#)

Create an Aggregate Storage Sample Using MaxL

You may wish to create an aggregate storage (ASO) version of a block storage (BSO) cube. The following sample MaxL script creates, loads data into, and aggregates an Essbase aggregate storage database that is based on Sample.Basic.

```

login $1 $2;

spool on to 'maxl_log.txt';

```

```
create or replace application SampleAS using aggregate_storage
comment 'aggregate storage version of Sample';

create database SampleAS.Basic2
comment 'aggregate storage version of Sample Basic';

create or replace outline on aggregate_storage database SampleAS.Basic2
as outline on database Sample.Basic;

alter database SampleAS.Basic2 initialize load buffer with buffer_id 1;

import database SampleAS.Basic2 data
from server data_file '/catalog/users/catalogUser/Data.txt'
to load_buffer with buffer_id 1
on error abort;

import database SampleAS.Basic2 data from load_buffer with buffer_id 1;

execute aggregate process on database SampleAS.Basic2
stopping when total_size exceeds 1.9;

spool off;

logout;
```

Related MaxL Links

[create application](#)

[create database](#)

[create outline](#)

[alter database \(for ASO\)](#)

[import data](#)

[execute aggregate process](#)

Load Data Using Buffers

The following example MaxL operations demonstrate how to use temporary load buffers to streamline data loading into an Essbase aggregate storage (ASO) database.

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 [List Aggregate Storage Data Load Buffers](#).

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.Basic
initialize load_buffer with buffer_id 1;
```

2. Load data into the buffer, using the [Import Data \(Aggregate Storage\)](#) statement.

```
import database ASOsamp.Basic data
from server data_file 'file_1'
to load_buffer with buffer_id 1
on error abort;
```

```
import database ASOsamp.Basic data
from server data_file 'file_2'
to load_buffer with buffer_id 1
on error abort;
```

```
import database ASOsamp.Basic 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.Basic 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.Basic
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.Basic
initialize load_buffer with buffer_id 1 resource_usage 0.5;
```

```
import database ASOsamp.Basic 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.Basic
initialize load_buffer with buffer_id 2 resource_usage 0.5;
```

```
import database ASOsamp.Basic 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.Basic data
from load_buffer with buffer_id 1, 2;
```

Related MaxL Links

[alter database \(for ASO\)](#)

[query database \(for ASO\)](#)

[import data \(for ASO\)](#)

List Aggregate Storage Data Load Buffers

The MaxL **query database** statement for ASO lists data load buffers that have been initialized to load data into an Essbase aggregate storage database.

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:

Table 3-24 List Load Buffers MaxL Output Columns

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.

Related MaxL Link

[query database \(for ASO\)](#)

Force Deletion of Partitions

If you need to drop an Essbase partition definition that is invalid, you can use the **force** keyword in the MaxL **drop partition** statement.

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

Metadata filtering enables an Essbase database administrator to remove certain outline members from a user's view. The MaxL create filter statement, indicating the MetaRead permission, is one way to set up a metadata 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 grid client 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. Oracle recommends 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:

Table 3-25 Sample Filter with 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:

Table 3-26 Sample Filter with No Overlap Conflicts

Access	Member Specification
MetaRead	California,West

Related MaxL Links[create filter](#)[alter filter](#)

Examples of Triggers

Using MaxL and MDX together, create and manage triggers that enable you to track state changes to an Essbase cube area.

The following examples are based on the Sample.Basic database.

**Note:**

You cannot define a trigger that requires data from Dynamic Calc 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 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@example.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.example.com],[sender@example.com],
      [recipient@example.com],
      [Subject of E-Mail])
end;
```

Related MaxL Links

[alter trigger](#)

[create trigger](#)

[display trigger](#)

[drop trigger](#)

4

Report Writer

Report Writer is a text-based script language that you can use to report on data in Essbase cubes. You can combine selection, layout, and formatting commands to build a variety of reports.

With Report Writer, you can generate reports whose length or specialized format exceed the capabilities of some grid clients. You can also use it to export data.

- [Report Writer Syntax](#)
- [Report Writer Command Groups](#)
- [Examples of Report Scripts](#)
- [Report Writer Command List](#)

Report Writer Syntax

Essbase Report Writer syntax uses delimiters to indicate commands, while other content is typically member names.

This topic contains the following information:

- [Report Delimiters](#)
- [Syntax Guidelines](#)
- [Referencing Static Members](#)

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 an Essbase member name.

Table 4-1 Report Writer Command Delimiters

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 Essbase Report Writer commands by at least one space. Terminate report scripts with an exclamation point. You can group commands, and insert descriptive code comments. Member names with spaces or special characters need to be enclosed in quotation marks.

- 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) Essbase member names, such as Sales and COGS, directly into report scripts. 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

Report Writer is a text-based script language that you can use to report on data in Essbase cubes. You can combine selection, layout, and formatting commands to build a variety of reports.

This section lists all Report Writer commands, grouped by command type. The command groups correspond to the steps of report design:

- [Report Layout Commands](#)
- [Data Range Commands](#)
- [Data Ordering Commands](#)
- [Member Selection and Sorting Commands](#)
- [Format Commands](#)
- [Column or Row Calculation Commands](#)
- [Member Names and Aliases](#)

Report Layout Commands

The Essbase report layout commands are listed below. These Report Writer commands provide column, page, and row layout options, and include two commands that override the default method for interpreting column dimension member lists.

- [ASYM](#)
- [COLUMN](#)
- [PAGE](#)
- [ROW](#)
- [SYM](#)

Data Range Commands

The Essbase Report Writer data range commands are listed below. These commands help you restrict the range of data selected for your reports.

- [BOTTOM](#)
- [RESTRICT](#)
- [TOP](#)

Data Ordering Commands

The Essbase Report Writer data ordering command, ORDERBY, helps you order data in your reports.

[ORDERBY](#)

Member Selection and Sorting Commands

The Essbase Report Writer member selection commands are listed below. These commands enhance your selection options using member relationships based on the database outline.

- [ALLINSAMEDIM](#)
- [ALLSIBLINGS](#)
- [ANCESTORS](#)
- [ATTRIBUTE](#)
- [CHILDREN](#)
- [CURRENCY](#)
- [DESCENDANTS](#)
- [DIMBOTTOM](#)
- [DIMEND](#)
- [DIMTOP](#)
- [DUPLICATE](#)
- [IANCESTORS](#)
- [ICCHILDREN](#)
- [IDESCENDANTS](#)
- [IPARENT](#)
- [LATEST](#)
- [LEAVES](#)
- [LINK](#)
- [MATCH](#)
- [OFSAMEGEN](#)
- [ONSAMELEVELAS](#)
- [PARENT](#)
- [SORTALTNAMES](#)
- [SORTASC](#)
- [SORTDESC](#)
- [SORTGEN](#)
- [SORTLEVEL](#)
- [SORTMBRNAMES](#)
- [SORTNONE](#)

- TODATE
- UDA
- WITHATTR

Format Commands

The Essbase Report Writer formatting commands are listed below. 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
- PAGELENGTH
- 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

The Essbase Report Writer column and row calculation commands are listed below. These commands 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

The Report Writer commands for working with Essbase member names and aliases are listed below. These commands 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, the following 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

The example Essbase Report Writer scripts in this section demonstrate report procedures and formats frequently required in business settings.

The samples use both the Demo Basic and Sample Basic cubes provided in the gallery.

The sample reports demonstrate the following techniques:

- [Sample 1: Creating a Different Format for Each Page](#)
- [Sample 2: Handling Missing Values](#)
- [Sample 3: Nesting Columns](#)
- [Sample 4: Grouping Rows](#)

- [Sample 5: Reporting on Different Combinations of Data](#)
- [Sample 6: Formatting Different Combinations of Data](#)
- [Sample 7: Using Aliases](#)
- [Sample 8: Creating Custom Headings and % Characters](#)
- [Sample 9: Creating Custom Page Headings](#)
- [Sample 10: Using Formulas](#)
- [Sample 11: Placing Two-Page Layouts on the Same Page](#)
- [Sample 12: Formatting for Data Export](#)
- [Sample 13: Creating Asymmetric Columns](#)
- [Sample 14: Calculating Columns](#)
- [Sample 15: Calculating Rows](#)
- [Sample 16: Sorting by Top or Bottom Data Values](#)
- [Sample 17: Restricting Rows](#)
- [Sample 18: Ordering Data Values](#)
- [Sample 19: Narrowing Member Selection Criteria](#)
- [Sample 20: Using Attributes in Member Selection](#)
- [Sample 21: Using the WITHATTR Command in Member Selection](#)

Sample 1: Creating a Different Format for Each Page

This Essbase report script generates a report for each page member.

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](#).

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](#) 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](#).

Sample 2: Handling Missing Values

This Essbase report script sample shows you how to use page breaks and labels for missing values.

This report has the same layout and member selection as Sample 1.

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" }
```

Sample 3: Nesting Columns

This Essbase report script sample demonstrates how to nest columns and suppress missing rows from the output.

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								
Qtr1	Jan		Feb		Mar			
	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.

Sample 4: Grouping Rows

This Essbase report script sample shows how to group rows to include two dimensions down the page.

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](#). 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

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
<CHILDREN Market
{ PAGEONDIMENSION Market }
{ SUPMISSINGROWS }

        <COLUMN (Scenario)
        <CHILDREN Scenario

<ROW(Product3, Year)
<CHILDREN Year
<IDESCENDANTS Product
!
```

Sample 5: Reporting on Different Combinations of Data

This Essbase report script sample demonstrates how to generate a multi page report on combinations of members across dimensions.

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](#) 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
<IDESCENDANTS 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
!
```

Sample 6: Formatting Different Combinations of Data

This Essbase report script sample shows how to format decimals, add underlining, and change how negative numbers display in reports.

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.

Sample 7: Using Aliases

This Essbase report script sample shows how to use aliases in a report, as well as how to suppress indentation, force a symmetric report, reorder columns, and restrict number of columns.

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	
<hr style="border-top: 1px dashed black;"/>					
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	
<hr style="border-top: 1px dashed black;"/>					
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.

Sample 8: Creating Custom Headings and % Characters

This Essbase report script sample shows how to create a custom header for a report, and include percent sign (%) characters after each data value.

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.

This report has custom headings at the top of each page. All format commands specified between the STARTHEADING and ENDHEADING format commands are displayed at the top of each report page.

TEXT format commands define text labels. The report generator provides *dynamic* text with **options*. This report uses the following options:

- *USERNAME, which outputs the user name used when connecting to Essbase Server
- *PAGESTRING, which outputs the current page number of the report
- C, which centers the report title

Sample 9: Creating Custom Page Headings

This Essbase report script sample shows how to create a custom header for a report.

This report builds on [Sample 8: Creating Custom Headings and % Characters](#) by adding custom page headings. By default, page dimension members are output at the top center of a report page. This section shows a representative part of the output.

```
Prepared by :admin                The Electronics Club                Page: 1
                                   Actual Profit by Product                12/12/01
```

Product: 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%
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%
Dallas	#Missing	#Missing	#Missing	#Missing	#Missing	#Missing
Houston	#Missing	#Missing	#Missing	#Missing	#Missing	#Missing
Phoenix	#Missing	#Missing	#Missing	#Missing	#Missing	#Missing
East	0.18%	-4.47%	-3.39%	-3.41%	-3.60%	-9.70%
West	0.98%	-2.95%	-0.13%	-2.81%	-2.62%	-5.61%
South	#Missing	#Missing	#Missing	#Missing	#Missing	#Missing
Market	0.65%	-3.56%	-1.44%	-3.06%	-3.03%	-7.29%

```
Prepared by :admin                The Electronics Club                Page: 2
                                   Actual Profit by Product                12/12/01
```

Product: 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%
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%
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                Page: 8
                                   Actual Profit by Product                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 to 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.

Sample 10: Using Formulas

This Essbase report script sample demonstrates how you can use column calculation formulas to 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](#), 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.

Sample 11: Placing Two-Page Layouts on the Same Page

This Essbase report script sample demonstrates how to build two different 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.

Sample 12: Formatting for Data Export

This report script 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 a 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.

Sample 13: Creating Asymmetric Columns

This Essbase report script sample makes asymmetric columns, and uses the RENAME command to avoid use of an alias table.

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 cube. In addition to allowing aliases, as in [Sample 7: Using Aliases](#), you can rename a row name in the report.

	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.

Sample 14: Calculating Columns

The following Essbase report script samples demonstrate use cases for CALCULATE COLUMN commands.

CALCULATE COLUMN supports standard mathematical operations.

- [Sample 14-A: Basic Calculated Columns](#)
- [Sample 14-B: Asymmetric Columns](#)

Sample 14-A: Basic Calculated Columns

This Essbase report script sample generates needed report columns using mathematical calculations.

```

                                     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)
```

(4)	25	27	14	66		Profit_%	26	22	22	70
(3)	56	62	59	177		Margin_%	60	60	60	180
(353)	1,417	1,120	898	3,435	VCR	Profit	1,466	1,161	1,161	3,788
(10)	33	30	24	87		Profit_%	35	31	31	98
(6)	61	61	62	183		Margin_%	63	63	63	189
(319)	400	272	256	928	Cam~	Profit	528	360	360	1,247
(10)	15	11	10	36		Profit_%	19	13	13	45
(2)	70	70	70	211		Margin_%	71	71	71	213
(832)	3,112	2,524	1,707	7,343	Visu~	Profit	3,234	2,471	2,471	8,175
(7)	25	24	17	66		Profit_%	27	23	23	74
(4)	61	63	63	187		Margin_%	64	64	64	191

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
  !
```

Sample 14-B: Asymmetric Columns

This Essbase report script sample generates two regular columns defined in asymmetric mode.

For an explanation of the use of asymmetric columns, see [Sample 13: Creating Asymmetric Columns](#).

East			
	Budget	Actual	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
!
```

Sample 15: Calculating Rows

The following Essbase report script samples demonstrate use cases for CALCULATE ROW commands.

- [Sample 15-A: Basic Calculated Row](#)

- [Sample 15-B: Calculated Rows and Missing Relationships](#)
- [Sample 15-C: Rows of Averages](#)

Sample 15-A: Basic Calculated Row

This Essbase report script sample 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
PRINTROW "Total Sales"
PRINTROW "Avg Sales" }
!
```

Sample 15-B: Calculated Rows and Missing Relationships

This Essbase report script summarizes information. When relationships that you need for reporting are not present in the outline, often the best solution is to use calculated rows (or columns).

This sample report is a simple summary of information in a North/South grouping, which is not part of the cube outline.

	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"
```

```
}  
!
```

Sample 15-C: Rows of Averages

This Essbase report script 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 cube 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
```

```
=====
```

		Television		VCR		Camera	
		Profit	Profit_%	Profit	Profit_%	Profit	Profit_%
		=====	=====	=====	=====	=====	=====
Qtr1							
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" }
!
```

Sample 16: Sorting by Top or Bottom Data Values

The following Essbase report script samples demonstrate the use of TOP and BOTTOM conditional retrieval commands.

- [Sample 16-A: Bottom Data Values](#)
- [Sample 16-B: Top Data Values](#)

Sample 16-A: Bottom Data Values

The following Essbase report script sample demonstrates the basic use of the BOTTOM command.

This sample 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.

Sample 16-B: Top Data Values

The following Essbase report script sample demonstrates the basic use of the TOP command.

This 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 following report script 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).

Sample 17: Restricting Rows

The following Essbase report script sample demonstrates how to use the RESTRICT conditional retrieval command in a report script.

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 (@DATACOLUMN(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.

Sample 18: Ordering Data Values

The following Essbase report script sample demonstrates how to use the ORDERBY conditional retrieval command in a report script.

		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", @DATACOLUMN(1) ASC, @DATACOLUMN(2) DESC, @DATACOLUMN(3)
ASC @DATACOLUMN (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

Sample 19: Narrowing Member Selection Criteria

The following Essbase report script sample demonstrates how to use the LINK command to narrow the members returned in a selection in a report script.

```
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.

Sample 20: Using Attributes in Member Selection

The following Essbase report script sample demonstrates how to use 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](#) for an example of substituting page breaks and labels for missing values.

Sample 21: Using the WITHATTR Command in Member Selection

The following Essbase report script sample demonstrates how to use 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.

Report Writer Command List

The following commands are available in Report Writer for Essbase.

Consult the Contents pane for a categorical list of Report Writer commands.

Table 4-2 Report Writer List

Alphabetical List of Report Writer Commands		
&	LINK	SAVEROW
!	LMARGIN	SCALE
ACCOFF	MASK	SETCENTER
ACCON	MATCH	SETROWOP
AFTER	MATCHEX	SINGLECOLUMN
ALLINSAMEDIM	MEANINGLESTEXT	SKIP
ALLSIBLINGS	MISSINGTEXT	SKIPONDIMENSION
ANCESTORS	NAMESCOL	SORTALTNAMES
ASYM	NAMESON	SORTASC
ATTRIBUTE	NAMEWIDTH	SORTDESC
ATTRIBUTEVA	NEWPAGE	SORTGEN
BEFORE	NOINDENTGEN	SORTLEVEL
BLOCKHEADERS	NOPAGEONDIMENSION	SORTMBRNames
BOTTOM	NOROWREPEAT	SORTNONE
BRACKETS	NOSKIPONDIMENSION	SPARSE
CALCULATE COLUMN	NOUNAMEONDIM	STARTHEADING
CALCULATE ROW	OFFCOLCALCS	SUDA
CHILDREN	OFFROWCALCS	SUPALL
CLEARALLROWCALC	OFSAMEGEN	SUPBRACKETS
CLEARROWCALC	ONCOLCALCS	SUPCOLHEADING
COLHEADING	ONROWCALCS	SUPCOMMAS
COLUMN	ONSAMELEVELAS	SUPCURHEADING
COMMAS	ORDER	SUPEMPTYROWS
CURHEADING	ORDERBY	SUPEUROPEAN
CURRENCY	OUTALT	SUPFEED
DATEFORMAT	OUTALTMBR	SUPFORMATS
DECIMAL	OUTALTNAMES	SUPHEADING
DESCENDANTS	OUTALTSELECT	SUPMASK
DIMBOTTOM	OUTFORMATTEDMISSING	SUPMISSINGROWS
DIMEND	OUTFORMATTEDVALUES	SUPNAMES
DIMTOP	OUTMBRALT	SUPOUTPUT
DUPLICATE	OUTMBRNames	SUPPAGEHEADING
ENDHEADING	OUTMEANINGLESS	SUPSHARE
EUROPEAN	OUTPUT	SUPSHAREOFF
FEEDON	OUTPUTMEMBERKEY	SUPZEROROWS

Table 4-2 (Cont.) Report Writer List

Alphabetical List of Report Writer Commands		
FIXCOLUMNS	PAGE	SYM
FORMATCOLUMNS	PAGEHEADING	TABDELIMIT
GEN	PAGELENGTH	TEXT
HEADING	PAGEONDIMENSION	TODATE
IANCESTORS	PARENT	TOP
ICHILDREN	PERSPECTIVE	UCHARACTERS
IDESCENDANTS	PRINTROW	UCOLUMNS
IMMHEADING	PYRAMIDHEADERS	UDA
INCEMPTYROWS	QUOTEMBRNAMES	UDATA
INCFORMATS	REMOVECOLCALCS	UNAME
INCMASK	RENAME	UNAMEONDIMENSION
INCMISSINGROWS	REPALIAS	UNDERLINECHAR
INCZEROROWS	REPALIASMBR	UNDERSCORECHAR
INDENT	REPMBR	WIDTH
INDENTGEN	REPMBRALIAS	WITHATTR
IPARENT	REPQUALMBR	WITHATTREX
LATEST	RESTRICT	ZEROTEXT
LEAVES	ROWREPEAT	
LEV	SAVEANDOUTPUT	

&

The Essbase Report Writer **&** command prefaces a substitution variable in the report script.

Syntax

& variableName

Parameters

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".

!

The Report Writer ! command 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

The Essbase Report Writer ACCOFF command turns off member accumulation (this is the default).

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

The following report script is designed for the Sample Basic cube, available in the gallery. <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

Related Topics

- [ACCON](#)
The Essbase Report Writer ACCON command turns on member accumulation.

ACCON

The Essbase Report Writer ACCON command 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

The following report script is designed for the Sample Basic cube, available in the gallery. The <ACCON command causes inclusion of all members, 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

Related Topics

- [ACCOFF](#)
The Essbase Report Writer ACCOFF command turns off member accumulation (this is the default).

AFTER

The Essbase Report Writer AFTER command 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] }
```

Parameters

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 *columnList*, 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 following report script is designed for the Demo Basic cube, available in the gallery. The {AFTER "%"} command 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%

Related Topics

- [BEFORE](#)
The Essbase Report Writer BEFORE command 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.

ALLINSAMEDIM

The Essbase Report Writer ALLINSAMEDIM command selects all the members from the same dimension as the specified dimension member for the report.

Syntax

```
<ALLINSAMEDIM mbrName
```

Parameters

mbrName

Single member representing a dimension. All members from this dimension are selected.

Example

The following report script is designed for the Demo Basic cube, available in the gallery.
<ALLINSAMEDIM Audio selects all the members from the dimension.

```
<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

Related Topics

- [ALLSIBLINGS](#)
The Essbase Report Writer ALLSIBLINGS command adds all the siblings of the specified member to the report.
- [DESCENDANTS](#)
The Essbase Report Writer DESCENDANTS command adds the descendants of *mbrName* to the report, excluding *mbrName*.

ALLSIBLINGS

The Essbase Report Writer ALLSIBLINGS command adds all the siblings of the specified member to the report.

Syntax

```
<ALLSIBLINGS mbrName
```

Parameters

mbrName

Name of member whose siblings you want to add.

Example

The following report script is designed for the Demo Basic cube, available in the gallery.
<ALLSIBLINGS Stereo selects the siblings of the member Stereo.

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual

<COLUMN (Year)
<ICHILDREN 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

Related Topics

- [ANCESTORS](#)
The Essbase Report Writer ANCESTORS command adds all the ancestors of the specified member to the report.
- [DESCENDANTS](#)
The Essbase Report Writer DESCENDANTS command adds the descendants of *mbrName* to the report, excluding *mbrName*.

ANCESTORS

The Essbase Report Writer ANCESTORS command adds all the ancestors of the specified member to the report.

Syntax

```
<ANCESTORS mbrName
```

Parameters

mbrName

Name of member whose ancestors you want to add.

Example

The following report script is designed for the Demo Basic cube, available in the gallery. <ANCESTORS Stereo adds Audio and Product to the report (as 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 script 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

Related Topics

- [IANCESTORS](#)
The Essbase Report Writer IANCESTORS command adds a member and its ancestors to the report.

ASYM

The Essbase Report Writer ASYM command 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 report script example is based on Sample Basic.

```
<PAGE (Measures, Market)
Texas Sales
<SYM
    <COLUMN (Scenario, Year)
    Actual Budget
    Jan Feb
<ROW (Product)
<IDESCENDANTS "100"
!
<ASYM
!
```

Which produces the following reports:

	Sales Texas		Budget	
	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

Related Topics

- [SYM](#)
The SYM command in Report Writer forces a symmetric report, regardless of the data selection in the report script. Use SYM to change the symmetry of a report that Essbase would otherwise create as an asymmetric report.

ATTRIBUTE

The Essbase Report Writer ATTRIBUTE command returns all base-dimension members associated with a specified attribute.

Syntax

```
<ATTRIBUTE attrMbrName
```

Parameters

attrMbrName

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 cube, `<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`).
- The 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.

The following report script is designed for the Sample Basic cube, available in the gallery. The script returns on rows only the names of the drinks that are associated with the member Ounces_12 on the corresponding attribute dimension.

```
<PAGE (Market, Measures, Scenario)  
    South Sales Actual
```

```
<COLUMN (Year)  
<ICHILDREN Year
```

```
{OUTALTNAMES}  
<ATTRIBUTE Ounces_12
```

```
!
```

The report script produces the following report:

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

Related Topics

- [WITHATTR](#)

The WITHATTR Report Writer command specifies the characteristics of a base-dimension member that match the specified values in a report script. You must create attribute dimensions in the Essbase outline, and associate them with a base dimension, before you use WITHATTR.

ATTRIBUTEVA

The Essbase Report Writer ATTRIBUTEVA command 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.

Syntax

```
<ATTRIBUTEVA (attrMbrName, options, startTuple[, endTuple])
```

Parameters

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))
```

Related Topics

- [WITHATTR](#)
The WITHATTR Report Writer command specifies the characteristics of a base-dimension member that match the specified values in a report script. You must create attribute dimensions in the Essbase outline, and associate them with a base dimension, before you use WITHATTR.
- [PERSPECTIVE](#)
The Essbase Report Writer PERSPECTIVE command sets the perspective, a tuple or REALITY, for a varying attribute dimension for a report.

BEFORE

The Essbase Report Writer BEFORE command 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 ] }
```

Parameters

char

A single-byte character enclosed in quotation marks.

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

The following report script is designed for the Demo Basic cube, available in the gallery. { BEFORE "\$" } is used to display the dollar sign before all the data values.

```
<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

Related Topics

- [AFTER](#)
The Essbase Report Writer AFTER command displays a character following the data columns in the report.

BLOCKHEADERS

The Essbase Report Writer BLOCKHEADERS command 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		Actual	
Budget		Budget	Budget	Budget	Budget	Budget	Budget	Budget	Budget	Budget	Budget
		Jan	Jan	Jan	Jan	Feb	Feb	Feb	Feb	Feb	Feb
Jan	Jan	Jan	Jan	Feb	Feb	Feb	Feb	Feb	Feb	Feb	Feb
		400-10	400-20	400-30	400	400-10	400-20	400-30	400		
400-10	400-20	400-30	400	400-10	400-20	400-30	400				
=====											
=====											
East		562	219	432	1,213	560	243	469	1,272		
580	230	440	1,250	580	260	490	1,330				
West		1,115	1,032	625	2,772	1,122	1,065	618	2,805		
740	690	410	1,840	740	700	400	1,840				

Related Topics

- [PYRAMIDHEADERS](#)

The Essbase Report Writer PYRAMIDHEADERS command displays column members in centered, pyramid-shaped levels above columns, which is the column display default style used by symmetric reports.

BOTTOM

The Essbase Report Writer BOTTOM command returns rows with the lowest values of a specified data column.

Syntax

```
<BOTTOM ([rowgroupDimension,] rows, column)
```

Parameters

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

@DATACOLUMN (colNumber) | @DATACOLUMN (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 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, <CHILDREN 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 configuration

Example

Example 1:

The following report script is designed for the Demo Basic cube, available in the gallery.

```
<Page (Market, Accounts, Scenario)
  Chicago Sales Actual
<Bottom (5, @DataColumn(4))
<Column(Year)
<Ichildren Year
<Row(Product)
<Idescendants Product
!
<Bottom (3, @DataColumn(1))
```

```
{Indentgen 3}
Boston Sales Actual
<Ichildren Year
<Idescendants Product
!
```

The report script 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
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

Example 2:

The following report script example is designed for the Sample Basic cube, available in the gallery. It uses the ORDERBY, TOP, BOTTOM, and RESTRICT functions:

```
<TOP ("Year", 10, @DataColumn(2))
{Width 15}
{Decimal 2}
{OutAltNames}
<BOTTOM ("Year", 5, @DataColumn(2))
<OutMBrAlt
<Column(Scenario)
{SupBrackets}
Actual Budget "Variance %"
<RESTRICT (@DataColumn(2) > 3000 and @DataColumn(1)
< 3500)
<Row(Year, Product)
<Idescendants Product
<Children Year
<OrderBy ( "Year",@DataColumn(1), @DataColumn(2) Desc)
!
```

The report script produces the following report:

Measures Market

			Actual	Budget	Variance
%			=====	=====	
=====					
Qtr1	300-10	Dark Cream	2,544.00	3,010.00	
-15.48					
	300-30	Diet Cream	2,695.00	3,070.00	
-12.21					
			2,695.00	3,070.00	
-12.21					
Qtr4	300-30	Diet Cream	2,820.00	3,080.00	
-8.44					
			2,820.00	3,080.00	
-8.44					
	200-20	Diet Root	2,834.00	3,790.00	
-25.22					
			2,834.00	3,790.00	
-25.22					
Qtr1	200-20	Diet Root	2,963.00	3,600.00	
-17.69					
			2,963.00	3,600.00	
-17.69					
Qtr2	200-20	Diet Root	3,079.00	3,640.00	
-15.41					
			3,079.00	3,640.00	
-15.41					
Qtr3	200-20	Diet Root	3,149.00	3,700.00	
-14.89					
			3,149.00	3,700.00	
-14.89					
Qtr2	300-10	Dark Cream	3,231.00	3,570.00	
-9.50					
Qtr3	300-10	Dark Cream	3,355.00	3,730.00	-10.05

Related Topics

- [RESTRICT](#)
The Essbase Report Writer RESTRICT command specifies the conditions that the row must satisfy before it becomes part of a result set.
- [TOP](#)
The TOP Report Writer command in Essbase returns rows with the highest values of a specified data column.
- [ORDERBY](#)
The Essbase Report Writer ORDERBY command orders the rows in a report according to data values in the specified columns.

BRACKETS

The Essbase Report Writer BRACKETS command 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.

Related Topics

- [SUPBRACKETS](#)

The Essbase Report Writer SUPBRACKETS command suppresses the display of parentheses around negative numbers.

CALCULATE COLUMN

The Essbase Report Writer CALCULATE COLUMN command 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 }
```

Parameters

"newColumn"

New column name enclosed by quotation marks.

expression

A column calculation expression.

If an operation or equation is not specified, the default is + (add).

The following mathematical operators are supported in column calculations:

- + 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 CALCULATE 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 Actual							
		Jan	Feb	Jan	Feb				
		400-10	400-20	400-20	400-30	400-10	400-30	Actual	Budget
		=====	=====	=====	=====	=====	=====	=====	=====
Market		2,839	2,562	2,596	1,233	2,879	1,261	5,401	4,112

		Sales Budget							
		Jan	Feb	Jan	Feb				
		400-10	400-20	400-20	400-30	400-10	400-30	Actual	Budget
		=====	=====	=====	=====	=====	=====	=====	=====
Market		2,320	2,040	2,050	990	2,350	1,030	4,360	3,340

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}
```

Related Topics

- [OFFCOLCALCS](#)
The Essbase Report Writer OFFCOLCALCS command disables all column calculations within the report.
- [ONCOLCALCS](#)
The Essbase Report Writer ONCOLCALCS command re-enables column calculations in the report after they have been disabled by OFFCOLCALCS.
- [REMOVECOLCALCS](#)
The Essbase Report Writer REMOVECOLCALCS command removes all column calculation definitions from the report.
- [SETROWOP](#)
The Essbase Report Writer SETROWOP command defines on-the-fly calculations for a named row created with CALCULATE ROW.

CALCULATE ROW

The Essbase Report Writer CALCULATE ROW command 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 ] }
```


Parameters

"newRow"

Name of a new row, enclosed by quotation marks, that was declared with SAVEROW or SAVEANDOUTPUT.

columnNo

Optional. Column numbers to which Essbase applies the expression.

expression

Row calculation expression. Member names are not supported.

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.
- CALCULATE ROW is limited to returning no more than 500 rows.
- 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.

Related Topics

- [CLEARROWCALC](#)
The Essbase Report Writer CLEARROWCALC command resets the value of the row calculation *name* to #MISSING.
- [CLEARALLROWCALC](#)
The Essbase Report Writer CLEARALLROWCALC command resets the value of all calculated rows to #MISSING.
- [DUPLICATE](#)
The Essbase Report Writer DUPLICATE command enables a member name to occur more than once in a dimension group selection.
- [OFFCOLCALCS](#)
The Essbase Report Writer OFFCOLCALCS command disables all column calculations within the report.
- [OFFROWCALCS](#)
The Essbase Report Writer OFFROWCALCS command temporarily disables all row calculations; for example, those calculations set by CALCULATE ROW.
- [ONCOLCALCS](#)
The Essbase Report Writer ONCOLCALCS command re-enables column calculations in the report after they have been disabled by OFFCOLCALCS.
- [ONROWCALCS](#)
The Essbase Report Writer ONROWCALCS command re-enables all row calculations after they have been disabled by OFFROWCALCS. Each subsequent row of data after using the command is calculated.
- [OUTPUT](#)
The Essbase Report Writer OUTPUT command resumes output, reversing the action of SUPOUTPUT.
- [PRINTROW](#)
The Essbase Report Writer PRINTROW command displays the calculated *rowName* with its current values.
- [REMOVECOLCALCS](#)
The Essbase Report Writer REMOVECOLCALCS command removes all column calculation definitions from the report.
- [RENAME](#)
The Essbase Report Writer RENAME command renames a member within the report.
- [SAVEROW](#)
The Essbase Report Writer SAVEROW command creates a new calculated row whose default name is specified by *rowMbr*, but which may be renamed with an optional name enclosed in quotation marks.
- [SETROWOP](#)
The Essbase Report Writer SETROWOP command defines on-the-fly calculations for a named row created with CALCULATE ROW.
- [SUPOUTPUT](#)
The Essbase Report Writer SUPOUTPUT command suppresses all output, except columns, while continuing to process other operations such as calculations or format settings. Use the OUTPUT command to resume output.

CHILDREN

The Essbase Report Writer CHILDREN command selects all members in the level immediately below the specified member, excluding the specified member.

Syntax

```
<CHILDREN mbrName
```

Parameters

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.

Related Topics

- [DESCENDANTS](#)
The Essbase Report Writer DESCENDANTS command adds the descendants of *mbrName* to the report, excluding *mbrName*.
- [ICCHILDREN](#)
The Essbase Report Writer ICHILDREN command selects the specified member and all members in the level immediately below it.
- [IDESCENDANTS](#)
The Essbase Report Writer IDESCENDANTS command adds the specified member and its descendants to the report.

CLEARALLROWCALC

The Essbase Report Writer CLEARALLROWCALC command resets the value of all calculated rows to #MISSING.

Syntax

```
{ CLEARALLROWCALC }
```

Related Topics

- [CALCULATE ROW](#)
The Essbase Report Writer CALCULATE ROW command 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.
- [CLEARROWCALC](#)
The Essbase Report Writer CLEARROWCALC command resets the value of the row calculation *name* to #MISSING.
- [OFFCOLCALCS](#)
The Essbase Report Writer OFFCOLCALCS command disables all column calculations within the report.
- [OFFROWCALCS](#)
The Essbase Report Writer OFFROWCALCS command temporarily disables all row calculations; for example, those calculations set by CALCULATE ROW.
- [ONCOLCALCS](#)
The Essbase Report Writer ONCOLCALCS command re-enables column calculations in the report after they have been disabled by OFFCOLCALCS.
- [ONROWCALCS](#)
The Essbase Report Writer ONROWCALCS command re-enables all row calculations after they have been disabled by OFFROWCALCS. Each subsequent row of data after using the command is calculated.
- [PRINTROW](#)
The Essbase Report Writer PRINTROW command displays the calculated *rowName* with its current values.
- [REMOVECOLCALCS](#)
The Essbase Report Writer REMOVECOLCALCS command removes all column calculation definitions from the report.
- [SETROWOP](#)
The Essbase Report Writer SETROWOP command defines on-the-fly calculations for a named row created with CALCULATE ROW.
- [SUPOUTPUT](#)
The Essbase Report Writer SUPOUTPUT command suppresses all output, except columns, while continuing to process other operations such as calculations or format settings. Use the OUTPUT command to resume output.

CLEARROWCALC

The Essbase Report Writer CLEARROWCALC command resets the value of the row calculation *name* to #MISSING.

Syntax

```
{ CLEARROWCALC name }
```

Parameters

name

Name of a calculated row from a CALCULATE ROW command.

Related Topics

- [CALCULATE ROW](#)
The Essbase Report Writer CALCULATE ROW command 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.
- [CLEARALLROWCALC](#)
The Essbase Report Writer CLEARALLROWCALC command resets the value of all calculated rows to #MISSING.
- [OFFCOLCALCS](#)
The Essbase Report Writer OFFCOLCALCS command disables all column calculations within the report.
- [OFFROWCALCS](#)
The Essbase Report Writer OFFROWCALCS command temporarily disables all row calculations; for example, those calculations set by CALCULATE ROW.
- [ONCOLCALCS](#)
The Essbase Report Writer ONCOLCALCS command re-enables column calculations in the report after they have been disabled by OFFCOLCALCS.
- [ONROWCALCS](#)
The Essbase Report Writer ONROWCALCS command re-enables all row calculations after they have been disabled by OFFROWCALCS. Each subsequent row of data after using the command is calculated.
- [PRINTROW](#)
The Essbase Report Writer PRINTROW command displays the calculated *rowName* with its current values.
- [REMOVECOLCALCS](#)
The Essbase Report Writer REMOVECOLCALCS command removes all column calculation definitions from the report.
- [RENAME](#)
The Essbase Report Writer RENAME command renames a member within the report.
- [SAVEANDOUTPUT](#)
The SAVEANDOUTPUT command in Essbase Report Writer adds a row member to the report, and creates a new calculated row whose default name is that row member. Its name can be changed using an optional row calculation name.

- **SAVEROW**
The Essbase Report Writer SAVEROW command creates a new calculated row whose default name is specified by *rowMbr*, but which may be renamed with an optional name enclosed in quotation marks.
- **SETROWOP**
The Essbase Report Writer SETROWOP command defines on-the-fly calculations for a named row created with CALCULATE ROW.
- **SUPOUTPUT**
The Essbase Report Writer SUPOUTPUT command suppresses all output, except columns, while continuing to process other operations such as calculations or format settings. Use the OUTPUT command to resume output.

COLHEADING

The Essbase Report Writer COLHEADING command 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 following report script is designed for the Demo Basic cube, available in the gallery. The COLHEADING command displays the column heading members for a second time after displaying a blank line with the SKIP command.

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual
  <COLUMN (Year)
    <CHILDREN Year
<ROW (Product)
<CHILDREN Audio
{ SKIP COLHEADING }
<CHILDREN 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

Related Topics

- [HEADING](#)
The Essbase Report Writer HEADING command displays the page heading: either the default heading, or the heading as defined with the STARTHEADING and ENDHEADING commands.
- [SUPCOLHEADING](#)
The SUPCOLHEADING command in Essbase Report Writer suppresses display of default column headings.
- [IMMHEADING](#)
The Essbase Report Writer IMMHEADING command forces the immediate display of the heading without waiting for the next non-suppressed data row.
- [SUPPAGEHEADING](#)
The Essbase Report Writer SUPPAGEHEADING command suppresses display of the page member heading whenever a heading is generated.
- [PAGEHEADING](#)
The Essbase Report Writer PAGEHEADING command displays the page heading before the next data-output row.

- **TEXT**
The TEXT Report Writer command in Essbase inserts text or other information on a new line in the report.

COLUMN

The Essbase Report Writer COLUMN command 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 )
```

Parameters

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.

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.

Related Topics

- **PAGE**
The Essbase Report Writer PAGE command defines which dimensions are displayed as page members in the final report.
- **ROW**
The Essbase Report Writer ROW command determines the row dimensions for a report whose member names appear in the data rows of the report.

COMMAS

The Essbase Report Writer **COMMAS** command 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.

Related Topics

- [BRACKETS](#)
The Essbase Report Writer **BRACKETS** command displays parentheses around negative numbers instead of negative signs.
- [DECIMAL](#)
The Essbase Report Writer **DECIMAL** command determines the number of decimal places to display in the report.
- [SUPALL](#)
The **SUPALL** command in Essbase Report Writer suppresses the display of the page and column headings, all member names, page breaks, commas, and brackets.
- [SUPCOMMAS](#)
The **SUPCOMMAS** command in Essbase Report Writer suppresses the display of commas in numbers greater than 999. The default behavior is to display the commas.

CURHEADING

The Essbase Report Writer **CURHEADING** command 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

Refer to the example for the [CURRENCY](#) command.

Related Topics

- [IMMHEADING](#)
The Essbase Report Writer IMMHEADING command forces the immediate display of the heading without waiting for the next non-suppressed data row.
- [CURRENCY](#)
The Essbase Report Writer CURRENCY command 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 cube: only in the report.
- [SUPCURHEADING](#)
The Essbase Report Writer SUPCURHEADING command suppresses the display of currency information when you use the CURRENCY command to convert the data values in your report to a specified currency.
- [TEXT](#)
The TEXT Report Writer command in Essbase inserts text or other information on a new line in the report.

CURRENCY

The Essbase Report Writer CURRENCY command 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 cube: only in the report.

If the <CURRENCY command is not used, the data is reported as it is currently stored in the cube.

Syntax

```
<CURRENCY targetCurrency
```

Parameters

targetCurrency

Currency and currency type to display in the report. Currency type is optional. Up to four members (at most one from each currency cube 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.

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
```

Related Topics

- [CURHEADING](#)
The Essbase Report Writer CURHEADING command enables the display of the currency conversion heading.
- [SUPCURHEADING](#)
The Essbase Report Writer SUPCURHEADING command suppresses the display of currency information when you use the CURRENCY command to convert the data values in your report to a specified currency.
- [TEXT](#)
The TEXT Report Writer command in Essbase inserts text or other information on a new line in the report.

DATEFORMAT

Report Writer can be used to prepare reports based on Date type members. Format directives that apply to numeric values also apply to Date type values. The {DATEFORMAT} format directive formats all the output cells based on the Essbase outline's date format string.

Report Writer post-processing commands that operate on numeric data, such as RESTRICT, TOP, BOTTOM and SORT, will operate on internal numeric date values.

Syntax

```
{ DATEFORMAT "string" }
```

Parameters

"string"

One of the date format strings supported by Essbase.

Related Topics

- [WITHATTR](#)
The WITHATTR Report Writer command specifies the characteristics of a base-dimension member that match the specified values in a report script. You must create attribute dimensions in the Essbase outline, and associate them with a base dimension, before you use WITHATTR.

DECIMAL

The Essbase Report Writer DECIMAL command determines the number of decimal places to display in the report.

Syntax

```
{ DECIMAL decPlaces | VARIABLE [ columnN [columnN] ] }
```

Parameters

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.

Related Topics

- [BRACKETS](#)
The Essbase Report Writer BRACKETS command displays parentheses around negative numbers instead of negative signs.
- [COMMAS](#)
The Essbase Report Writer COMMAS command displays commas for numbers greater than 999 after commas have been suppressed with either a SUPCOMMAS or SUPALL command.
- [SUPBRACKETS](#)
The Essbase Report Writer SUPBRACKETS command suppresses the display of parentheses around negative numbers.

- **SUPCOMMAS**
The SUPCOMMAS command in Essbase Report Writer suppresses the display of commas in numbers greater than 999. The default behavior is to display the commas.

DESCENDANTS

The Essbase Report Writer DESCENDANTS command 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]])
```

Parameters

mbrName

Name of parent of descendants.

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

- 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 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

Related Topics

- [CHILDREN](#)
The Essbase Report Writer CHILDREN command selects all members in the level immediately below the specified member, excluding the specified member.
- [ICHILDREN](#)
The Essbase Report Writer ICHILDREN command selects the specified member and all members in the level immediately below it.
- [IDESCENDANTS](#)
The Essbase Report Writer IDESCENDANTS command adds the specified member and its descendants to the report.
- [LINK](#)
The Essbase Report Writer LINK command uses the AND, OR, and NOT Boolean operators, combined with extraction commands, to refine member selections.

DIMBOTTOM

The Essbase Report Writer DIMBOTTOM command adds all level 0 dimension members to the report.

Syntax

```
<DIMBOTTOM mbrName
```

Parameters

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

Related Topics

- [DIMTOP](#)
The Essbase Report Writer DIMTOP command adds the top of the dimension for the member to the report.

DIMEND

The Essbase Report Writer DIMEND command 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
```

Parameters

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

The Essbase Report Writer DIMTOP command adds the top of the dimension for the member to the report.

Syntax

```
<DIMTOP mbrName
```

Parameters

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

Related Topics

- [DIMBOTTOM](#)
The Essbase Report Writer DIMBOTTOM command adds all level 0 dimension members to the report.
- [DIMEND](#)
The Essbase Report Writer DIMEND command specifies a dimension format to be processed after cycling through all members in the dimension.

DUPLICATE

The Essbase Report Writer DUPLICATE command enables a member name to occur more than once in a dimension group selection.

The DUPLICATE command is useful either of the following cases:

- in a multi-section report when the same row name appears more than once in each section.
- 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
```

Parameters

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 Demo Basic.

```

<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:

East

Budget Jan =====			Actual Jan =====	Actual % 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%

Related Topics

- [PAGE](#)
The Essbase Report Writer PAGE command defines which dimensions are displayed as page members in the final report.
- [COLUMN](#)
The Essbase Report Writer COLUMN command defines the dimensions displayed as column members. Column members are displayed above data columns.
- [ROW](#)
The Essbase Report Writer ROW command determines the row dimensions for a report whose member names appear in the data rows of the report.

ENDHEADING

The Essbase Report Writer ENDHEADING command 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.

Related Topics

- **HEADING**
The Essbase Report Writer HEADING command displays the page heading: either the default heading, or the heading as defined with the STARTHEADING and ENDHEADING commands.
- **IMMHEADING**
The Essbase Report Writer IMMHEADING command forces the immediate display of the heading without waiting for the next non-suppressed data row.
- **STARTHEADING**
The Essbase Report Writer STARTHEADING 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.
- **SUPHEADING**
The Essbase Report Writer SUPHEADING command suppresses the display of the default heading (page header and column headers) or custom header, if defined, at the top of each page.

EUROPEAN

The Essbase Report Writer EUROPEAN command 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 Demo Basic.

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

```

Related Topics

- [BRACKETS](#)
The Essbase Report Writer BRACKETS command displays parentheses around negative numbers instead of negative signs.
- [COMMAS](#)
The Essbase Report Writer COMMAS command displays commas for numbers greater than 999 after commas have been suppressed with either a SUPCOMMAS or SUPALL command.
- [DECIMAL](#)
The Essbase Report Writer DECIMAL command determines the number of decimal places to display in the report.
- [SUPBRACKETS](#)
The Essbase Report Writer SUPBRACKETS command suppresses the display of parentheses around negative numbers.
- [SUPCOMMAS](#)
The SUPCOMMAS command in Essbase Report Writer suppresses the display of commas in numbers greater than 999. The default behavior is to display the commas.
- [SUPEUROPEAN](#)
The Essbase Report Writer SUPEUROPEAN command disables the European method for displaying numbers.

FEEDON

The Essbase Report Writer FEEDON command 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.

Related Topics

- [PAGELENGTH](#)
The Essbase Report Writer PAGELENGTH command sets the maximum number of lines for one page in the report.
- [SUPFEED](#)
The Essbase Report Writer SUPFEED command suppresses the automatic insertion of a physical page break whenever the number of lines on a page exceeds the current PAGELENGTH setting.

FIXCOLUMNS

The Essbase Report Writer FIXCOLUMNS command fixes the number of columns in the report regardless of how many columns are originally selected.

Syntax

```
{ FIXCOLUMNS number }
```

Parameters

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 report script examples are designed for the Demo Basic cube, available in the gallery.

In the following example, { 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)
```



```

<ICHILDREN Year

<ROW (Product)

{FIXCOLUMNS 3}
<ICHILDREN 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

<ICHILDREN 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
```

Note that without the FIXCOLUMNS, the column headers would have been:

```

      Jan      Feb      Mar
Actual Budget Actual Budget Actual Budget
```

Related Topics

- [ORDER](#)
The Essbase Report Writer ORDER command specifies the order of columns in a report, based on the original ordering of the columns.

FORMATCOLUMNS

The Essbase Report Writer FORMATCOLUMNS command expands the number of data columns when processed.

Syntax

```
{ FORMATCOLUMNS number }
```

Parameters**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.

Related Topics

- [COLUMN](#)
The Essbase Report Writer COLUMN command defines the dimensions displayed as column members. Column members are displayed above data columns.
- [NAMECOL](#)
The Essbase Report Writer NAMECOL command determines the location of the row names columns in the report.

GEN

The Essbase Report Writer GEN command returns all members in a dimension with the specified generation name.

Syntax 1

```
GEN name, dimension
```

Syntax 2

When used as an extraction command in conjunction with the <LINK command, the syntax is:

```
<GEN(dimension, genNumber)
```

Parameters

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.

Related Topics

- [LEV](#)
The Essbase Report Writer LEV command returns all members in a dimension with the specified level name.

- [LINK](#)
The Essbase Report Writer LINK command uses the AND, OR, and NOT Boolean operators, combined with extraction commands, to refine member selections.

HEADING

The Essbase Report Writer HEADING command 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 and the column member 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.

Related Topics

- [COLHEADING](#)
The Essbase Report Writer COLHEADING command 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.
- [ENDHEADING](#)
The Essbase Report Writer ENDHEADING command ends the definition of the custom page heading displayed at the top of each page.
- [IMMHEADING](#)
The Essbase Report Writer IMMHEADING command forces the immediate display of the heading without waiting for the next non-suppressed data row.
- [PAGEHEADING](#)
The Essbase Report Writer PAGEHEADING command displays the page heading before the next data-output row.
- [STARTHEADING](#)
The Essbase Report Writer STARTHEADING 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.
- [SUPHEADING](#)
The Essbase Report Writer SUPHEADING command suppresses the display of the default heading (page header and column headers) or custom header, if defined, at the top of each page.

IANCESTORS

The Essbase Report Writer IANCESTORS command adds a member and its ancestors to the report.

Syntax

```
<IANCESTORS mbrName
```

Parameters

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.

Related Topics

- [CHILDREN](#)
The Essbase Report Writer CHILDREN command selects all members in the level immediately below the specified member, excluding the specified member.
- [DESCENDANTS](#)
The Essbase Report Writer DESCENDANTS command adds the descendants of *mbrName* to the report, excluding *mbrName*.

- **PARENT**
The Essbase Report Writer PARENT command adds the parent of the specified member to the report.

ICHILDREN

The Essbase Report Writer ICHILDREN command selects the specified member and all members in the level immediately below it.

Syntax

```
<ICHILDREN mbrName
```

Parameters

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.

Related Topics

- **ANCESTORS**
The Essbase Report Writer ANCESTORS command adds all the ancestors of the specified member to the report.
- **CHILDREN**
The Essbase Report Writer CHILDREN command selects all members in the level immediately below the specified member, excluding the specified member.
- **DESCENDANTS**
The Essbase Report Writer DESCENDANTS command adds the descendants of *mbrName* to the report, excluding *mbrName*.
- **PARENT**
The Essbase Report Writer PARENT command adds the parent of the specified member to the report.

IDESCENDANTS

The Essbase Report Writer IDESCENDANTS command adds the specified member and its descendants to the report.

Syntax 1

```
<IDESCENDANTS mbrName
```

Syntax 2

When used as an extraction command in conjunction with the <LINK command, the syntax is:

```
<IDESCENDANTS (mbrName [, gen|levelName [, AT|UPTO]])
```

Parameters

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

The following report script is designed for the Demo Basic cube, available in the gallery.

<IDESCENDANTS Product adds all the members from the Product dimension to the report. 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

The following report script is designed for the Demo Basic cube, available in the gallery.

```
<LINK(<IDESCENDANTS(Market,"Lev0,Market"))
!
```

This example produces the following report:

Year Product Accounts Scenario	
New_York	15,647
Boston	15,644
Chicago	15,285
San_Francisco	20,727
Seattle	14,667
Denver	13,016
Los_Angeles	14,429
Dallas	8,585
Houston	7,874
Phoenix	8,106
Market	133,980

Example 3

The following report script is designed for the Demo Basic cube, available in the gallery.

```
<LINK(<IDESCENDANTS(Market,"Lev0,Market",UPTO))
!
```

This example produces the following report:

Year Product Accounts Scenario	
Market	133,980
New_York	15,647
Boston	15,644

Chicago	15,285
East	46,576
San_Francisco	20,727
Seattle	14,667
Denver	13,016
Los_Angeles	14,429
West	62,839
Dallas	8,585
Houston	7,874
Phoenix	8,106
South	24,565

Related Topics

- [ANCESTORS](#)
The Essbase Report Writer ANCESTORS command adds all the ancestors of the specified member to the report.
- [CHILDREN](#)
The Essbase Report Writer CHILDREN command selects all members in the level immediately below the specified member, excluding the specified member.
- [DESCENDANTS](#)
The Essbase Report Writer DESCENDANTS command adds the descendants of *mbrName* to the report, excluding *mbrName*.
- [PARENT](#)
The Essbase Report Writer PARENT command adds the parent of the specified member to the report.
- [LINK](#)
The Essbase Report Writer LINK command uses the AND, OR, and NOT Boolean operators, combined with extraction commands, to refine member selections.

IMMHEADING

The Essbase Report Writer IMMHEADING command 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.

Related Topics

- [ENDHEADING](#)
The Essbase Report Writer ENDHEADING command ends the definition of the custom page heading displayed at the top of each page.
- [HEADING](#)
The Essbase Report Writer HEADING command displays the page heading: either the default heading, or the heading as defined with the STARTHEADING and ENDHEADING commands.
- [STARTHEADING](#)
The Essbase Report Writer STARTHEADING 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.
- [SUPHEADING](#)
The Essbase Report Writer SUPHEADING command suppresses the display of the default heading (page header and column headers) or custom header, if defined, at the top of each page.

INCEMPTYROWS

The Essbase Report Writer INCEMPTYROWS command displays empty rows of data, or rows that contain only zeros or #MISSING data values, in the final report.

Syntax

```
{ INCEMPTYROWS }
```

Notes

INCEMPTYROWS cancels the effects of SUPEMPTYROWS, SUPMISSINGROWS or SUPZEROROWS.

Related Topics

- [INCMISSINGROWS](#)
The Essbase Report Writer INCMISSINGROWS command displays missing rows of data, or rows that contain all #MISSING data values, in the final report.
- [INCZEROROWS](#)
The Essbase Report Writer INCZEROROWS command includes rows that contain only data values of zero in the final report.
- [SUPALL](#)
The SUPALL command in Essbase Report Writer suppresses the display of the page and column headings, all member names, page breaks, commas, and brackets.
- [SUPEMPTYROWS](#)
The Essbase Report Writer SUPEMPTYROWS command suppresses the display of empty rows – that is, rows that have only 0 or #MISSING values in the row.

- [SUPMISSINGROWS](#)
The Essbase Report Writer SUPMISSINGROWS command can control the display of empty values that are #MISSING in the report. It suppresses the display of rows that contain only #MISSING values.
- [SUPZEROROWS](#)
The Essbase Report Writer SUPZEROROWS command is one of the commands that helps deal with empty values in a report. It suppresses the display of rows that have only 0 values.

INCFORMATS

The Essbase Report Writer INCFORMATS command 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.

Related Topics

- [SUPFORMATS](#)
The Essbase Report Writer SUPFORMATS command suppresses formats that produce extra output, such as underlines and line skips.

INCMASK

The Essbase Report Writer INCMASK command re-includes (turns back on) the mask that has been suppressed by the command SUPMASK.

Syntax

```
{ INCMASK }
```

Related Topics

- [MASK](#)
The Essbase Report Writer MASK command overwrites the text in each output row with the specified characters at the specified position.

INCMISSINGROWS

The Essbase Report Writer INCMISSINGROWS command 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.

Related Topics

- [INCEMPTYROWS](#)
The Essbase Report Writer INCEMPTYROWS command displays empty rows of data, or rows that contain only zeros or #MISSING data values, in the final report.
- [INCZEROROWS](#)
The Essbase Report Writer INCZEROROWS command includes rows that contain only data values of zero in the final report.
- [SUPALL](#)
The SUPALL command in Essbase Report Writer suppresses the display of the page and column headings, all member names, page breaks, commas, and brackets.
- [SUPEMPTYROWS](#)
The Essbase Report Writer SUPEMPTYROWS command suppresses the display of empty rows – that is, rows that have only 0 or #MISSING values in the row.
- [SUPMISSINGROWS](#)
The Essbase Report Writer SUPMISSINGROWS command can control the display of empty values that are #MISSING in the report. It suppresses the display of rows that contain only #MISSING values.
- [SUPZEROROWS](#)
The Essbase Report Writer SUPZEROROWS command is one of the commands that helps deal with empty values in a report. It suppresses the display of rows that have only 0 values.

INCZEROROWS

The Essbase Report Writer INCZEROROWS command includes rows that contain only data values of zero in the final report.

Syntax

```
{ INCZEROROWS }
```

Notes

Use INCZEROROWS to re-include rows containing zeroes after a SUPZEROROWS or SUPEMPTYROWS command has been used to suppress the zero rows from the report.

Related Topics

- [INCEMPTYROWS](#)
The Essbase Report Writer INCEMPTYROWS command displays empty rows of data, or rows that contain only zeros or #MISSING data values, in the final report.
- [INCMISSINGROWS](#)
The Essbase Report Writer INCMISSINGROWS command displays missing rows of data, or rows that contain all #MISSING data values, in the final report.
- [SUPALL](#)
The SUPALL command in Essbase Report Writer suppresses the display of the page and column headings, all member names, page breaks, commas, and brackets.
- [SUPEMPTYROWS](#)
The Essbase Report Writer SUPEMPTYROWS command suppresses the display of empty rows – that is, rows that have only 0 or #MISSING values in the row.
- [SUPMISSINGROWS](#)
The Essbase Report Writer SUPMISSINGROWS command can control the display of empty values that are #MISSING in the report. It suppresses the display of rows that contain only #MISSING values.
- [SUPZEROROWS](#)
The Essbase Report Writer SUPZEROROWS command is one of the commands that helps deal with empty values in a report. It suppresses the display of rows that have only 0 values.

INDENT

The Essbase Report Writer INDENT command shifts the first row names column in column-output order by the specified number of characters. The default indentation, if no offset is specified, is 2.

Syntax

```
{ INDENT [ offset ] }
```

Parameters

offset

Optional. Number of spaces to indent column 0 from the left boundary of the name column.
Values:

- 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

The following report script is designed for the Demo Basic cube, available in the gallery. The first report script for Chicago generates 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)
<CHILDREN 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

Related Topics

- [INDENTGEN](#)
The Essbase Report Writer INDENTGEN command indents subsequent row members in the row names column based on the generation in the cube outline.
- [LMARGIN](#)
The Essbase Report Writer LMARGIN command sets the left margin for the report to a specified number of characters.
- [NOINDENTGEN](#)
The Essbase Report Writer NOINDENTGEN command displays all row member names left-aligned in the row names column without indenting members based on generation in the database outline.

INDENTGEN

The Essbase Report Writer INDENTGEN command indents subsequent row members in the row names column based on the generation in the cube outline.

Syntax

```
{ INDENTGEN [ offset ] }
```

Parameters

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 cube outline. Generations are counted starting, from 1, 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 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 report script is designed for the Demo Basic cube, available in the gallery. The Chicago report uses the default generation indentation, followed by the Boston report, which uses {INDENTGEN 3}.

```
<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

Related Topics

- [INDENT](#)
The Essbase Report Writer INDENT command shifts the first row names column in column-output order by the specified number of characters. The default indentation, if no offset is specified, is 2.
- [NOINDENTGEN](#)
The Essbase Report Writer NOINDENTGEN command displays all row member names left-aligned in the row names column without indenting members based on generation in the database outline.

IPARENT

The Essbase Report Writer IPARENT command adds the specified member and its parent to the report.

Syntax

```
IPARENT mbrName
```

Parameters**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.

Related Topics

- [PARENT](#)
The Essbase Report Writer PARENT command adds the parent of the specified member to the report.

LATEST

The Essbase Report Writer LATEST command 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.)

Syntax 1

If you use the following syntax, the LATEST command is applied globally in the report script.

```
<LATEST mbrName
```

Syntax 2

If you use the following syntax, Essbase applies the LATEST command only to the member listed in the syntax argument.

```
<LATEST reservedName (mbrName)
```

Parameters

reservedName

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)

mbrName

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

The Essbase Report Writer LEAVES command 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
```

Parameters

mbrName

Single member whose level 0 contributing descendants should be added to the report

Notes

- LEAVES command only applies to aggregate storage databases.
- LEAVES command cannot be used to rename a member in a report script.
- LEAVES command can only be used on rows or pages; if used on columns, an error is returned.
- LEAVES command is not supported in combination with name and alias sorting commands. Members will be returned in outline order.
- LEAVES command is not supported in combination with other selection commands for the same dimension.
- LEAVES 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".

Related Topics

- [DESCENDANTS](#)
The Essbase Report Writer DESCENDANTS command adds the descendants of *mbrName* to the report, excluding *mbrName*.

LEV

The Essbase Report Writer LEV command returns all members in a dimension with the specified level name.

Syntax 1

```
LEV name, dimension
```

Syntax 2

When used as an extraction command in conjunction with the <LINK command, the syntax is:

```
<LEV(dimension, levNumber)
```

Parameters

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.

Related Topics

- [GEN](#)
The Essbase Report Writer GEN command returns all members in a dimension with the specified generation name.
- [LINK](#)
The Essbase Report Writer LINK command uses the AND, OR, and NOT Boolean operators, combined with extraction commands, to refine member selections.

LINK

The Essbase Report Writer LINK command uses the AND, OR, and NOT Boolean operators, combined with extraction commands, to refine member selections.

Syntax

```
<LINK (extractionCommand [operator extractionCommand])
```

Parameters

extractionCommand

Any of the following extraction commands or another AND/OR expression:

- <ALLINSAMEDIM (*member*)
- <ALLSIBLINGS (*member*)
- <ANCESTORS (*member*)
- <CHILDREN (*member*)
- <DESCENDANTS (*member* [, *gen/levelName* [, AT|UPTO]])
- <DIMBOTTOM (*member*)
- <DIMTOP (*member*)
- <IANCESTORS (*member*)
- <ICHILDREN (*member*)
- <IDESCENDANTS (*member* [, *gen/levelName* [, AT|UPTO]])
- <IPARENT (*member*)
- <MATCH (*Dimension*, *match_string*)
- <MEMBER (*member*)
- <OFSAMEGEN (*member*)

- <ONSAMELEVELAS (*member*)
- <PARENT (*member*)
- <UDA (*Dimension, UDA_name*)

Operator

Any of the following Boolean operators:

- 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.
- If the LINK command returns an empty set of members, nothing is returned.

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.

<LINK (<DESCENDANTS (Market, "Lev0,Market")

OR

(<DESCENDANTS (Market, "State"))

!

returns the following report (works on Sample Basic cube):

Year Measures	Product	Scenario
New York	8,202	
Massachusetts	6,712	
Florida	5,029	
Connecticut	3,093	
New Hampshire	1,125	
California	12,964	
Oregon	5,062	
Washington	4,641	
Utah	3,155	
Nevada	4,039	
Texas	6,425	
Oklahoma	3,491	
Louisiana	2,992	
New Mexico	330	
Illinois	12,577	
Ohio	4,384	
Wisconsin	3,547	
Missouri	1,466	
Iowa	9,061	
Colorado	7,227	

Related Topics

- [ALLINSAMEDIM](#)
The Essbase Report Writer ALLINSAMEDIM command selects all the members from the same dimension as the specified dimension member for the report.
- [ALLSIBLINGS](#)
The Essbase Report Writer ALLSIBLINGS command adds all the siblings of the specified member to the report.
- [ANCESTORS](#)
The Essbase Report Writer ANCESTORS command adds all the ancestors of the specified member to the report.
- [CHILDREN](#)
The Essbase Report Writer CHILDREN command selects all members in the level immediately below the specified member, excluding the specified member.

- **DESCENDANTS**
The Essbase Report Writer DESCENDANTS command adds the descendants of *mbrName* to the report, excluding *mbrName*.
- **DIMBOTTOM**
The Essbase Report Writer DIMBOTTOM command adds all level 0 dimension members to the report.
- **DIMTOP**
The Essbase Report Writer DIMTOP command adds the top of the dimension for the member to the report.
- **IANCESTORS**
The Essbase Report Writer IANCESTORS command adds a member and its ancestors to the report.
- **ICHILDREN**
The Essbase Report Writer ICHILDREN command selects the specified member and all members in the level immediately below it.
- **IDESCENDANTS**
The Essbase Report Writer IDESCENDANTS command adds the specified member and its descendants to the report.
- **IPARENT**
The Essbase Report Writer IPARENT command adds the specified member and its parent to the report.
- **MATCH**
The MATCH Report Writer command performs wildcard member selection. Essbase searches for member names that match the pattern you specify, and returns the member names it finds.
- **OFSAMEGEN**
The Essbase Report Writer OFSAMEGEN command adds to the report the members from the same dimension and generation as the specified member.
- **ONSAMELEVELAS**
The Essbase Report Writer ONSAMELEVELAS command adds to the report all members on the same level as the specified member.
- **PARENT**
The Essbase Report Writer PARENT command adds the parent of the specified member to the report.
- **UDA**
The UDA Report Writer command in Essbase selects and reports on members based on a common attribute, defined as a UDA (user-defined attribute).

LMARGIN

The Essbase Report Writer LMARGIN command sets the left margin for the report to a specified number of characters.

Syntax

```
{ LMARGIN [ marginSize ] }
```


Parameters

marginSize

Optional numeric value: number of character spaces for left margin.

Notes

LMARGIN 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 LMARGIN 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.

Related Topics

- [INDENT](#)
The Essbase Report Writer INDENT command shifts the first row names column in column-output order by the specified number of characters. The default indentation, if no offset is specified, is 2.
- [PAGELENGTH](#)
The Essbase Report Writer PAGELENGTH command sets the maximum number of lines for one page in the report.

MASK

The Essbase Report Writer MASK command overwrites the text in each output row with the specified characters at the specified position.

All non-blank 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.

Table 4-3 MASK Keywords

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.

Table 4-3 (Cont.) MASK Keywords

Keyword	Description
*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.
*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" ] }
```

Parameters**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 the Sample Basic cube.

```
<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

Related Topics

- [INCMASK](#)
The Essbase Report Writer INCMASK command re-includes (turns back on) the mask that has been suppressed by the command SUPMASK.
- [SUPMASK](#)
The Essbase Report Writer SUPMASK command suppresses the display of a text mask.
- [TEXT](#)
The TEXT Report Writer command in Essbase inserts text or other information on a new line in the report.

MATCH

The MATCH Report Writer command performs wildcard member selection. Essbase searches for member names that match the pattern you specify, and returns the member names it finds.

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")
```

Parameters

"Member"

Member name at the top of the member hierarchy you want to search. Essbase searches the member name and its descendants.

If the client is set to use aliases in place of member names, the MATCH command searches for alias names.

"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 ?).

- ? 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 ("").

Example

The following report is based on the Sample Basic cube, 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 cube, 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

The MATCHEX Report Writer command performs wildcard member selection. Essbase searches for member names that match the pattern you specify, and returns the member names it finds.

You can optionally 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.

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)
```

Parameters

"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 ?).

- ? 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.

- **ALT**
Filter using aliases of selected members from selected alias table for pattern matching. The alias table is set by `outaltselect`, 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 `outaltselect`, otherwise default alias table.

Example

The following report script example is designed for the Sample Basic cube, available in the gallery.

```
<OUTALTSELECT default
<MATCHEX(Product,Caff*,ALT)
!
<OUTALTSELECT "Default"
<NewAlt "Product"
<OUTMBRNAME
<LINK(<MATCHEX("Product", "100", MBR) AND <IDESCENDANTS("Product"))
!
```

This example produces the following report:

Year	Measures	Market	Scenario
100-30		1,983	
Colas		30,468	

MEANINGLESSTEXT

The Essbase Report Writer **MEANINGLESSTEXT** command displays `#ME` in reports for text strings you specify. Use it with `OUTMEANINGLESS` to mark cells that are meaningless because no base member/attribute member combination exists.

Syntax

```
{ MEANINGLESSTEXT "string" }
```

Parameters**"string"**

The specified string to be replaced with `#ME` in cells.

Related Topics

- [WITHATTR](#)
The **WITHATTR** Report Writer command specifies the characteristics of a base-dimension member that match the specified values in a report script. You must create attribute

dimensions in the Essbase outline, and associate them with a base dimension, before you use WITHATTR.

MISSINGTEXT

The Essbase Report Writer MISSINGTEXT command replaces the default #MISSING label with your specified text, when a missing data value is generated on a line in the report.

Syntax

```
{MISSINGTEXT [ "text" ] }
```

Parameters

text

Optional text to use for missing values.

Notes

- MISSINGTEXT is a report setting command.
- The text label you specify must be enclosed in double quotes.

Example

```
{MISSINGTEXT "Not Applicable."}
```

Related Topics

- [SUEMPTYROWS](#)
The Essbase Report Writer SUEMPTYROWS command suppresses the display of empty rows – that is, rows that have only 0 or #MISSING values in the row.
- [SUPMISSINGROWS](#)
The Essbase Report Writer SUPMISSINGROWS command can control the display of empty values that are #MISSING in the report. It suppresses the display of rows that contain only #MISSING values.
- [SUPZEROROWS](#)
The Essbase Report Writer SUPZEROROWS command is one of the commands that helps deal with empty values in a report. It suppresses the display of rows that have only 0 values.
- [TEXT](#)
The TEXT Report Writer command in Essbase inserts text or other information on a new line in the report.

NAMESCOL

The Essbase Report Writer NAMESCOL command determines the location of the row names columns in the report.

Use the NAMESCOL command *after* entering the column members in the report. You can get the same result with the ORDER command, but NAMESCOL is more convenient for moving just the names columns and when the number of data columns can vary.

Syntax

```
{ NAMESCOL [ columnList | CENTERED ] }
```

Parameters

columnList

Optional list, separated by spaces, of the locations for each row name. List position corresponds to the number of the affected column.

NAMESCOL 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:

- Define all columns in the report.
- Use the FORMATCOLUMNS command to set the number of columns.

Notes

{ NAMESCOL 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 following report script is designed for the Demo Basic cube, available in the gallery.

{ NAMESCOL 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)
{ NAMESCOL 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

Related Topics

- [FIXCOLUMNS](#)
The Essbase Report Writer FIXCOLUMNS command fixes the number of columns in the report regardless of how many columns are originally selected.
- [FORMATCOLUMNS](#)
The Essbase Report Writer FORMATCOLUMNS command expands the number of data columns when processed.
- [NAMEWIDTH](#)
The Essbase Report Writer NAMEWIDTH command determines the width of all row name columns in the report.
- [ORDER](#)
The Essbase Report Writer ORDER command specifies the order of columns in a report, based on the original ordering of the columns.

NAMESON

The Essbase Report Writer NAMESON command 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.

Related Topics

- [SUPALL](#)
The SUPALL command in Essbase Report Writer suppresses the display of the page and column headings, all member names, page breaks, commas, and brackets.
- [SUPNAMES](#)
The Essbase Report Writer SUPNAMES command suppresses the display of row member names in the final report.

NAMEWIDTH

The Essbase Report Writer NAMEWIDTH command determines the width of all row name columns in the report.

Syntax

```
{ NAMEWIDTH [ width ] }
```

Parameters**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

The following report script is designed for the Demo Basic cube, available in the gallery. 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

Related Topics

- [NAMESCOL](#)
The Essbase Report Writer NAMESCOL command determines the location of the row names columns in the report.
- [WIDTH](#)
The WIDTH Report Writer command in Essbase specifies the width of columns in a report.

NEWPAGE

The Essbase Report Writer NEWPAGE command 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.

Related Topics

- [FEEDON](#)
The Essbase Report Writer FEEDON command enables page break insertion when the number of output lines on a page is greater than the PAGELENGTH setting.
- [SUPFEED](#)
The Essbase Report Writer SUPFEED command suppresses the automatic insertion of a physical page break whenever the number of lines on a page exceeds the current PAGELENGTH setting.

NOINDENTGEN

The Essbase Report Writer NOINDENTGEN command 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.

Related Topics

- [INDENT](#)
The Essbase Report Writer INDENT command shifts the first row names column in column-output order by the specified number of characters. The default indentation, if no offset is specified, is 2.
- [INDENTGEN](#)
The Essbase Report Writer INDENTGEN command indents subsequent row members in the row names column based on the generation in the cube outline.

NOPAGEONDIMENSION

The Essbase Report Writer NOPAGEONDIMENSION 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.

Syntax

```
{ NOPAGEONDIMENSION mbrName }
```

Parameters**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.

Related Topics

- [NOSKIPONDIMENSION](#)
The Essbase Report Writer NOSKIPONDIMENSION command prevents insertion of a new line when a member from the same dimension as the input member changes in a row of the report.
- [PAGEONDIMENSION](#)
The Essbase Report Writer PAGEONDIMENSION command 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.
- [SKIPONDIMENSION](#)
The Essbase Report Writer SKIPONDIMENSION command inserts a blank line when a member from the same dimension as the specified member changes on the next line in the report.

NOROWREPEAT

The Essbase Report Writer NOROWREPEAT 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. 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

Because NOROWREPEAT is the default, you need only use this command after using ROWREPEAT.

Example

The following example is based on the Demo Basic cube, available in the gallery of applications in the Essbase file catalog.

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

Related Topics

- [ROWREPEAT](#)
The Essbase Report Writer ROWREPEAT command 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.

NOSKIPONDIMENSION

The Essbase Report Writer NOSKIPONDIMENSION command prevents insertion of a new line when a member from the same dimension as the input member changes in a row of the report.

Syntax

```
{ NOSKIPONDIMENSION mbrName }
```

Parameters**mbrName**

Single member that defines a dimension for which to halt line-skipping.

Notes

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.

Related Topics

- [NOPAGEONDIMENSION](#)
The Essbase Report Writer NOPAGEONDIMENSION 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.
- [PAGEONDIMENSION](#)
The Essbase Report Writer PAGEONDIMENSION command 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.
- [SKIPONDIMENSION](#)
The Essbase Report Writer SKIPONDIMENSION command inserts a blank line when a member from the same dimension as the specified member changes on the next line in the report.

NOUNAMEONDIM

The Essbase Report Writer NOUNAMEONDIM command 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 }
```

Parameters

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.

Related Topics

- [NOPAGEONDIMENSION](#)
The Essbase Report Writer NOPAGEONDIMENSION 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.
- [NOSKIPONDIMENSION](#)
The Essbase Report Writer NOSKIPONDIMENSION command prevents insertion of a new line when a member from the same dimension as the input member changes in a row of the report.
- [PAGEONDIMENSION](#)
The Essbase Report Writer PAGEONDIMENSION command 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.
- [SKIPONDIMENSION](#)
The Essbase Report Writer SKIPONDIMENSION command inserts a blank line when a member from the same dimension as the specified member changes on the next line in the report.

- [UNAMEONDIMENSION](#)
The UNAMEONDIMENSION Report Writer command in Essbase underlines the row member names in a row whenever a member from the same dimension as the specified member changes.

OFFCOLCALCS

The Essbase Report Writer OFFCOLCALCS command disables all column calculations within the report.

Syntax

```
{ OFFCOLCALCS }
```

Notes

OFFCOLCALCS 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.

Related Topics

- [CALCULATE COLUMN](#)
The Essbase Report Writer CALCULATE COLUMN command creates a new report column, performs on-the-fly calculations, and displays the calculation results in the newly-created column.
- [CLEARROWCALC](#)
The Essbase Report Writer CLEARROWCALC command resets the value of the row calculation *name* to #MISSING.
- [CLEARALLROWCALC](#)
The Essbase Report Writer CLEARALLROWCALC command resets the value of all calculated rows to #MISSING.
- [OFFROWCALCS](#)
The Essbase Report Writer OFFROWCALCS command temporarily disables all row calculations; for example, those calculations set by CALCULATE ROW.
- [ONCOLCALCS](#)
The Essbase Report Writer ONCOLCALCS command re-enables column calculations in the report after they have been disabled by OFFCOLCALCS.
- [ONROWCALCS](#)
The Essbase Report Writer ONROWCALCS command re-enables all row calculations after they have been disabled by OFFROWCALCS. Each subsequent row of data after using the command is calculated.
- [PRINTROW](#)
The Essbase Report Writer PRINTROW command displays the calculated *rowName* with its current values.
- [REMOVECOLCALCS](#)
The Essbase Report Writer REMOVECOLCALCS command removes all column calculation definitions from the report.

- [SETROWOP](#)
The Essbase Report Writer SETROWOP command defines on-the-fly calculations for a named row created with CALCULATE ROW.

OFFROWCALCS

The Essbase Report Writer OFFROWCALCS command temporarily disables all row calculations; for example, those calculations set by CALCULATE ROW.

Syntax

```
{ OFFROWCALCS }
```

Notes

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.

Related Topics

- [CALCULATE ROW](#)
The Essbase Report Writer CALCULATE ROW command 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.
- [CLEARROWCALC](#)
The Essbase Report Writer CLEARROWCALC command resets the value of the row calculation *name* to #MISSING.
- [CLEARALLROWCALC](#)
The Essbase Report Writer CLEARALLROWCALC command resets the value of all calculated rows to #MISSING.
- [OFFCOLCALCS](#)
The Essbase Report Writer OFFCOLCALCS command disables all column calculations within the report.
- [ONCOLCALCS](#)
The Essbase Report Writer ONCOLCALCS command re-enables column calculations in the report after they have been disabled by OFFCOLCALCS.
- [ONROWCALCS](#)
The Essbase Report Writer ONROWCALCS command re-enables all row calculations after they have been disabled by OFFROWCALCS. Each subsequent row of data after using the command is calculated.
- [PRINTROW](#)
The Essbase Report Writer PRINTROW command displays the calculated *rowName* with its current values.
- [REMOVECOLCALCS](#)
The Essbase Report Writer REMOVECOLCALCS command removes all column calculation definitions from the report.

- [SETROWOP](#)
The Essbase Report Writer SETROWOP command defines on-the-fly calculations for a named row created with CALCULATE ROW.

OFSAMEGEN

The Essbase Report Writer OFSAMEGEN command adds to the report the members from the same dimension and generation as the specified member.

Syntax

```
<OFSAMEGEN mbrName
```

Parameters

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

The following report script is designed for the Demo Basic cube, available in the gallery.

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual

<COLUMN (Year)
<ICHILDREN 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

Related Topics

- [ALLINSAMEDIM](#)
The Essbase Report Writer ALLINSAMEDIM command selects all the members from the same dimension as the specified dimension member for the report.

- **CHILDREN**
The Essbase Report Writer CHILDREN command selects all members in the level immediately below the specified member, excluding the specified member.
- **DESCENDANTS**
The Essbase Report Writer DESCENDANTS command adds the descendants of *mbrName* to the report, excluding *mbrName*.
- **ONSAMELEVELAS**
The Essbase Report Writer ONSAMELEVELAS command adds to the report all members on the same level as the specified member.

ONCOLCALCS

The Essbase Report Writer ONCOLCALCS command re-enables column calculations in the report after they have been disabled by OFFCOLCALCS.

Syntax

```
{ ONCOLCALCS }
```

Example

See the example for the [CALCULATE COLUMN](#) command.

Related Topics

- **CALCULATE COLUMN**
The Essbase Report Writer CALCULATE COLUMN command creates a new report column, performs on-the-fly calculations, and displays the calculation results in the newly-created column.
- **CLEARROWCALC**
The Essbase Report Writer CLEARROWCALC command resets the value of the row calculation *name* to #MISSING.
- **CLEARALLROWCALC**
The Essbase Report Writer CLEARALLROWCALC command resets the value of all calculated rows to #MISSING.
- **OFFCOLCALCS**
The Essbase Report Writer OFFCOLCALCS command disables all column calculations within the report.
- **OFFROWCALCS**
The Essbase Report Writer OFFROWCALCS command temporarily disables all row calculations; for example, those calculations set by CALCULATE ROW.
- **ONROWCALCS**
The Essbase Report Writer ONROWCALCS command re-enables all row calculations after they have been disabled by OFFROWCALCS. Each subsequent row of data after using the command is calculated.
- **PRINTROW**
The Essbase Report Writer PRINTROW command displays the calculated *rowName* with its current values.
- **REMOVECOLCALCS**
The Essbase Report Writer REMOVECOLCALCS command removes all column calculation definitions from the report.

- [SETROWOP](#)
The Essbase Report Writer SETROWOP command defines on-the-fly calculations for a named row created with CALCULATE ROW.

ONROWCALCS

The Essbase Report Writer ONROWCALCS command 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 }
```

Example

See the example for the [CALCULATE ROW](#) command.

Related Topics

- [CALCULATE ROW](#)
The Essbase Report Writer CALCULATE ROW command 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.
- [CLEARROWCALC](#)
The Essbase Report Writer CLEARROWCALC command resets the value of the row calculation *name* to #MISSING.
- [CLEARALLROWCALC](#)
The Essbase Report Writer CLEARALLROWCALC command resets the value of all calculated rows to #MISSING.
- [OFFCOLCALCS](#)
The Essbase Report Writer OFFCOLCALCS command disables all column calculations within the report.
- [ONCOLCALCS](#)
The Essbase Report Writer ONCOLCALCS command re-enables column calculations in the report after they have been disabled by OFFCOLCALCS.
- [REMOVECOLCALCS](#)
The Essbase Report Writer REMOVECOLCALCS command removes all column calculation definitions from the report.

ONSAMELEVELAS

The Essbase Report Writer ONSAMELEVELAS command adds to the report all members on the same level as the specified member.

Syntax

```
<ONSAMELEVELAS mbrName
```

Parameters

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 cube 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

The following report script is designed for the Demo Basic cube, available in the gallery.

```

<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
Visual	10,795	10,102	11,812	14,365	47,074

Related Topics

- [ALLINSAMEDIM](#)
The Essbase Report Writer ALLINSAMEDIM command selects all the members from the same dimension as the specified dimension member for the report.
- [CHILDREN](#)
The Essbase Report Writer CHILDREN command selects all members in the level immediately below the specified member, excluding the specified member.
- [DESCENDANTS](#)
The Essbase Report Writer DESCENDANTS command adds the descendants of *mbrName* to the report, excluding *mbrName*.
- [OFSAMEGEN](#)
The Essbase Report Writer OFSAMEGEN command adds to the report the members from the same dimension and generation as the specified member.

ORDER

The Essbase Report Writer ORDER command 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 }
```

Parameters

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

Related Topics

- [FIXCOLUMNS](#)
The Essbase Report Writer FIXCOLUMNS command fixes the number of columns in the report regardless of how many columns are originally selected.
- [NAMESCOL](#)
The Essbase Report Writer NAMESCOL command determines the location of the row names columns in the report.

ORDERBY

The Essbase Report Writer ORDERBY command orders the rows in a report according to data values in the specified columns.

Syntax

```
<ORDERBY ( [rowgroupDimension], column [direction]{,column
[direction]})
```

Parameters

<Optional rowgroup Dimension>

Row grouping dimension that determines the rows to sort as a set.

<column>

@DATACOLUMN (<colnumber>) | @DATACOLUMN (<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 <DESCENDANTS).

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

The following report script is designed for the Sample Basic cube, available in the gallery.

```
//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", @DATACOLUMN(1) ASC, @DATACOLUMN(2) DESC, @DATACOLUMN(3)
ASC)
!
// End of report
```

The report script above produces the following report:

		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

Related Topics

- [RESTRICT](#)
The Essbase Report Writer RESTRICT command specifies the conditions that the row must satisfy before it becomes part of a result set.
- [TOP](#)
The TOP Report Writer command in Essbase returns rows with the highest values of a specified data column.

- **BOTTOM**
The Essbase Report Writer BOTTOM command returns rows with the lowest values of a specified data column.

OUTALT

The OUTALT Report Writer command sets the output alias to the Essbase 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

Related Topics

- [OUTALTMBR](#)
The Essbase Report Writer OUTALTMBR command sets the output alias to the alias name (as defined in the current alias table in the outline) followed by the member name.
- [OUTALTNAMES](#)
The Essbase Report Writer OUTALTNAMES command displays alias names for members in a report.
- [OUTMBRALT](#)
The OUTMBRALT Report Writer command for Essbase sets the output name as the member name followed by the alias, as defined in the current alias table. The member name and alias are separated by a single space.
- [OUTMBRNAMES](#)
The Essbase Report Writer OUTMBRNAMES command reverts to the default member name display after the OUTALTNAMES command has been used to display alternate names. The member name is the default name used for reports.

OUTALTMBR

The Essbase Report Writer OUTALTMBR command sets the output alias to the alias name (as defined in the current alias table in the outline) followed by the 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 {OUTALTNAMES} command. If no alternate name exists, only the member name is displayed.
- OUTALTMBR cannot be used on duplicate member outlines. See [REPALIASMBR](#).

Example

The following example is based on Sample Basic.

```
<PAGE (Product, Measures)
<COLUMN (Scenario, Year)
{OUTALTNAMES}
<OUTALTMBR
Actual
<CHILDREN Qtr1
<ROW (Market)
<IDESCENDANTS "300"
!
```

The report script example above 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

Related Topics

- [OUTALT](#)
The OUTALT Report Writer command sets the output alias to the Essbase alias name, as defined in the current alias table.
- [OUTALTNAMES](#)
The Essbase Report Writer OUTALTNAMES command displays alias names for members in a report.
- [OUTMBRALT](#)
The OUTMBRALT Report Writer command for Essbase sets the output name as the member name followed by the alias, as defined in the current alias table. The member name and alias are separated by a single space.
- [REPALIASMBR](#)
The Essbase Report Writer REPALIASMBR command displays alias names followed by member names for members of the dimension specified in the report output.

OUTALTNAMES

The Essbase Report Writer OUTALTNAMES command displays alias names for members in a report.

OUTALTNAMES can 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"  
{OUTMBRNAME}
```

```
// 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

Related Topics

- [OUTALT](#)
The OUTALT Report Writer command sets the output alias to the Essbase alias name, as defined in the current alias table.
- [OUTALTMBR](#)
The Essbase Report Writer OUTALTMBR command sets the output alias to the alias name (as defined in the current alias table in the outline) followed by the member name.
- [OUTMBRALT](#)
The OUTMBRALT Report Writer command for Essbase sets the output name as the member name followed by the alias, as defined in the current alias table. The member name and alias are separated by a single space.
- [OUTMBRNames](#)
The Essbase Report Writer OUTMBRNames command reverts to the default member name display after the OUTALTNAMES command has been used to display alternate names. The member name is the default name used for reports.

OUTALTSELECT

The Essbase Report Writer OUTALTSELECT command 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
```

Parameters

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"
!
```

Related Topics

- [REPALIAS](#)
The Essbase Report Writer REPALIAS command displays alias names for members of the dimension specified.
- [REPALIASMBR](#)
The Essbase Report Writer REPALIASMBR command displays alias names followed by member names for members of the dimension specified in the report output.
- [REPMBR](#)
The Essbase Report Writer REPMBR command displays member names only for members of the dimension specified.
- [REPMBRALIAS](#)
The Essbase Report Writer REPMBRALIAS command displays member names followed by aliases for members of the dimension specified.
- [OUTALTMBR](#)
The Essbase Report Writer OUTALTMBR command sets the output alias to the alias name (as defined in the current alias table in the outline) followed by the member name.
- [OUTALTNAMES](#)
The Essbase Report Writer OUTALTNAMES command displays alias names for members in a report.

- [OUTMBRALT](#)
The OUTMBRALT Report Writer command for Essbase sets the output name as the member name followed by the alias, as defined in the current alias table. The member name and alias are separated by a single space.
- [OUTMBRNames](#)
The Essbase Report Writer OUTMBRNames command reverts to the default member name display after the OUTALTNAMES command has been used to display alternate names. The member name is the default name used for reports.

OUTFORMATTEDMISSING

The Essbase Report Writer OUTFORMATTEDMISSING command 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 }
```

Related Topics

- [WITHATTR](#)
The WITHATTR Report Writer command specifies the characteristics of a base-dimension member that match the specified values in a report script. You must create attribute dimensions in the Essbase outline, and associate them with a base dimension, before you use WITHATTR.

OUTFORMATTEDVALUES

The OUTFORMATTEDVALUES Report Writer command for Essbase 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 }
```

Related Topics

- [WITHATTR](#)
The WITHATTR Report Writer command specifies the characteristics of a base-dimension member that match the specified values in a report script. You must create attribute dimensions in the Essbase outline, and associate them with a base dimension, before you use WITHATTR.

OUTMBRALT

The OUTMBRALT Report Writer command for Essbase sets the output name as the member name followed by the 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 OUTALT NAMES 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)
{OUTALT NAMES}
<OUTMBRALT
Actual
<CHILDREN Qtr1
<ROW (Market)
<IDESCENDANTS "300"
!
```

The above report script 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

Related Topics

- [OUTALT](#)
The OUTALT Report Writer command sets the output alias to the Essbase alias name, as defined in the current alias table.
- [OUTALTMBR](#)
The Essbase Report Writer OUTALTMBR command sets the output alias to the alias name (as defined in the current alias table in the outline) followed by the member name.
- [OUTALTNAMES](#)
The Essbase Report Writer OUTALTNAMES command displays alias names for members in a report.
- [OUTMBRNAMES](#)
The Essbase Report Writer OUTMBRNAMES command reverts to the default member name display after the OUTALTNAMES command has been used to display alternate names. The member name is the default name used for reports.
- [REPMBRALIAS](#)
The Essbase Report Writer REPMBRALIAS command displays member names followed by aliases for members of the dimension specified.

OUTMBRNAMES

The Essbase Report Writer OUTMBRNAMES command reverts to the default member name display after the OUTALTNAMES command has been used to display alternate names. The member name is the default name used for reports.

Syntax

```
{ OUTMBRNAMES }
```

Notes

OUTMBRNAMES cannot be used on duplicate member outlines. See [REPMBR](#).

Related Topics

- [OUTALT](#)
The OUTALT Report Writer command sets the output alias to the Essbase alias name, as defined in the current alias table.
- [OUTALTMBR](#)
The Essbase Report Writer OUTALTMBR command sets the output alias to the alias name (as defined in the current alias table in the outline) followed by the member name.
- [OUTALTNAMES](#)
The Essbase Report Writer OUTALTNAMES command displays alias names for members in a report.
- [OUTMBRALT](#)
The OUTMBRALT Report Writer command for Essbase sets the output name as the member name followed by the alias, as defined in the current alias table. The member name and alias are separated by a single space.

OUTMEANINGLESS

The Essbase Report Writer OUTMEANINGLESS command displays #ME in reports for cells that are meaningless because no base member-attribute member combination exists.

Syntax

```
{ OUTMEANINGLESS }
```

Related Topics

- [WITHATTR](#)
The WITHATTR Report Writer command specifies the characteristics of a base-dimension member that match the specified values in a report script. You must create attribute dimensions in the Essbase outline, and associate them with a base dimension, before you use WITHATTR.

OUTPUT

The Essbase Report Writer OUTPUT command 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.

Related Topics

- [SUPOUTPUT](#)
The Essbase Report Writer SUPOUTPUT command suppresses all output, except columns, while continuing to process other operations such as calculations or format settings. Use the OUTPUT command to resume output.

OUTPUTMEMBERKEY

The Essbase Report Writer OUTPUTMEMBERKEY command 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.

Related Topics

- [REPQUALMBR](#)
For the Essbase dimension specified, the Report Writer REPQUALMBR command displays member names for any unique member names, and displays a system generated identifier (a qualified name) for any duplicate member names. REPQUALMBR applies to duplicate member outlines only.
- [REPMBR](#)
The Essbase Report Writer REPMBR command displays member names only for members of the dimension specified.
- [REPALIAS](#)
The Essbase Report Writer REPALIAS command displays alias names for members of the dimension specified.
- [REPMBRALIAS](#)
The Essbase Report Writer REPMBRALIAS command displays member names followed by aliases for members of the dimension specified.
- [REPALIASMBR](#)
The Essbase Report Writer REPALIASMBR command displays alias names followed by member names for members of the dimension specified in the report output.

PAGE

The Essbase Report Writer PAGE command defines which dimensions are displayed as page members in the final report.

PAGE 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 )
```

Parameters

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.
- 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

The following example 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.

```
<PAGE (Measures, Market)
```

Related Topics

- [COLUMN](#)
The Essbase Report Writer COLUMN command defines the dimensions displayed as column members. Column members are displayed above data columns.
- [ROW](#)
The Essbase Report Writer ROW command determines the row dimensions for a report whose member names appear in the data rows of the report.

PAGEHEADING

The Essbase Report Writer PAGEHEADING command displays the page heading before the next data-output row.

Without PAGEHEADING, 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 following report script is designed for the Demo Basic cube, available in the gallery. The PAGEHEADING command 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

Related Topics

- [COLHEADING](#)
The Essbase Report Writer COLHEADING command 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.
- [HEADING](#)
The Essbase Report Writer HEADING command displays the page heading: either the default heading, or the heading as defined with the STARTHEADING and ENDHEADING commands.
- [PAGE](#)
The Essbase Report Writer PAGE command defines which dimensions are displayed as page members in the final report.
- [SUPALL](#)
The SUPALL command in Essbase Report Writer suppresses the display of the page and column headings, all member names, page breaks, commas, and brackets.
- [SUPCOLHEADING](#)
The SUPCOLHEADING command in Essbase Report Writer suppresses display of default column headings.
- [SUPHEADING](#)
The Essbase Report Writer SUPHEADING command suppresses the display of the default heading (page header and column headers) or custom header, if defined, at the top of each page.
- [SUPPAGEHEADING](#)
The Essbase Report Writer SUPPAGEHEADING command suppresses display of the page member heading whenever a heading is generated.
- [TEXT](#)
The TEXT Report Writer command in Essbase inserts text or other information on a new line in the report.

PAGELENGTH

The Essbase Report Writer PAGELENGTH command sets the maximum number of lines for one page in the report.

Syntax

```
{ PAGELENGTH [ lines ] }
```

Parameters

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 the *lines* parameter is not given.

After displaying the specified 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 printing, for legal size paper, the value should be 84 lines. For A4 paper, the value should be 70 lines.

Example

```
{ PAGELENGTH 50 } sets the maximum number of lines for one page to 50.
```

Related Topics

- [LMARGIN](#)
The Essbase Report Writer LMARGIN command sets the left margin for the report to a specified number of characters.
- [WIDTH](#)
The WIDTH Report Writer command in Essbase specifies the width of columns in a report.

PAGEONDIMENSION

The Essbase Report Writer PAGEONDIMENSION command 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 }
```

Parameters

mbrName

Single member. If any member of the same dimension increments, a new page is started.

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.

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, changing dimensions.

When combined with UNAMEONDIMENSION and SKIPONDIMENSION, UNAMEONDIMENSION is processed first, followed by SKIPONDIMENSION and PAGEONDIMENSION, in that order.

Example

In the following report script example, the command { PAGEONDIMENSION Year } inserts a page break before displaying the members Qtr2, Qtr3, and Qtr4. 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
```

Related Topics

- [NOPAGEONDIMENSION](#)
The Essbase Report Writer NOPAGEONDIMENSION 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.
- [NOSKIPONDIMENSION](#)
The Essbase Report Writer NOSKIPONDIMENSION command prevents insertion of a new line when a member from the same dimension as the input member changes in a row of the report.
- [SKIPONDIMENSION](#)
The Essbase Report Writer SKIPONDIMENSION command inserts a blank line when a member from the same dimension as the specified member changes on the next line in the report.

PARENT

The Essbase Report Writer PARENT command adds the parent of the specified member to the report.

Syntax

```
<PARENT mbrName
```

Parameters

mbrName

Single member, which must not be the dimension (top) member.

Example

The following command adds Qtr1 to the report.

```
<PARENT Jan
```

Related Topics

- [ANCESTORS](#)
The Essbase Report Writer ANCESTORS command adds all the ancestors of the specified member to the report.
- [CHILDREN](#)
The Essbase Report Writer CHILDREN command selects all members in the level immediately below the specified member, excluding the specified member.
- [DESCENDANTS](#)
The Essbase Report Writer DESCENDANTS command adds the descendants of *mbrName* to the report, excluding *mbrName*.

PERSPECTIVE

The Essbase Report Writer PERSPECTIVE command sets the perspective, a tuple or REALITY, for a varying attribute dimension for a report.

Syntax

```
<PERSPECTIVE(tuple, attrDim)
```

Parameters

tuple

(*m1*, *m2*, ..., *mX*) | REALITY

The perspective tuple to be applied for the given attribute dimension.

- (*m1*, *m2*, ..., *mN*)

Level-0 members from one or more independent dimensions for *attrDim* 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)
```

Related Topics

- [WITHATTREX](#)
The `WITHATTREX` Report Writer command 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 Essbase outline, and associate them with a base dimension, before you use `WITHATTREX` in a report script.
- [ATTRIBUTEVA](#)
The Essbase Report Writer `ATTRIBUTEVA` command 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.

PRINTROW

The Essbase Report Writer `PRINTROW` command displays the calculated `rowName` with its current values.

Syntax

```
{ PRINTROW "rowName" }
```

Parameters

"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.

Related Topics

- [CALCULATE COLUMN](#)
The Essbase Report Writer [CALCULATE COLUMN](#) command creates a new report column, performs on-the-fly calculations, and displays the calculation results in the newly-created column.
- [CLEARROWCALC](#)
The Essbase Report Writer [CLEARROWCALC](#) command resets the value of the row calculation *name* to #MISSING.
- [CLEARALLROWCALC](#)
The Essbase Report Writer [CLEARALLROWCALC](#) command resets the value of all calculated rows to #MISSING.
- [OFFCOLCALCS](#)
The Essbase Report Writer [OFFCOLCALCS](#) command disables all column calculations within the report.
- [OFFROWCALCS](#)
The Essbase Report Writer [OFFROWCALCS](#) command temporarily disables all row calculations; for example, those calculations set by [CALCULATE ROW](#).
- [ONCOLCALCS](#)
The Essbase Report Writer [ONCOLCALCS](#) command re-enables column calculations in the report after they have been disabled by [OFFCOLCALCS](#).
- [ONROWCALCS](#)
The Essbase Report Writer [ONROWCALCS](#) command re-enables all row calculations after they have been disabled by [OFFROWCALCS](#). Each subsequent row of data after using the command is calculated.
- [REMOVECOLCALCS](#)
The Essbase Report Writer [REMOVECOLCALCS](#) command removes all column calculation definitions from the report.
- [RENAME](#)
The Essbase Report Writer [RENAME](#) command renames a member within the report.
- [SAVEANDOUTPUT](#)
The [SAVEANDOUTPUT](#) command in Essbase Report Writer adds a row member to the report, and creates a new calculated row whose default name is that row member. Its name can be changed using an optional row calculation name.
- [SAVEROW](#)
The Essbase Report Writer [SAVEROW](#) command creates a new calculated row whose default name is specified by *rowMbr*, but which may be renamed with an optional name enclosed in quotation marks.

- **SETROWOP**
The Essbase Report Writer SETROWOP command defines on-the-fly calculations for a named row created with CALCULATE ROW.

PYRAMIDHEADERS

The Essbase Report Writer PYRAMIDHEADERS command displays column members in centered, pyramid-shaped levels above columns, which is the column display default style used by symmetric reports.

Syntax

```
{PYRAMIDHEADERS}
```

Notes

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

PYRAMIDHEADERS is the default behavior for symmetric reports. This command can be used to reset to the default column display style following a BLOCKHEADERS command.

Example

The following report script 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

Related Topics

- [BLOCKHEADERS](#)
The Essbase Report Writer BLOCKHEADERS command displays all members that apply to a column as the column heading, in the style used by asymmetric reports.

QUOTEMBRNAMES

The Essbase Report Writer QUOTEMBRNAMES command displays all the member names within quotation marks in the report script output.

Note that when the report script is run through Smart View or another grid client, 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 cube 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

The following report script is designed for the Sample Basic cube, available in the gallery.

```
<PAGE (Scenario)
<COLUMN (Year)
<ROW (Product, Market, Measures)
<QUOTEMBRNAMES
{ROWREPEAT}

<CHILDREN Year
<DIMBOTTOM Product
<DIMBOTTOM Market
<CHILDREN Profit
!
```

which produces the following report (truncated in this example):

		"Scenario"			
"Qtr4"	"Year"	"Qtr1"	"Qtr2"	"Qtr3"	
=====	=====	=====	=====	=====	
"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				
...					

REMOVECOLCALCS

The Essbase Report Writer REMOVECOLCALCS command removes all column calculation definitions from the report.

Syntax

```
{ REMOVECOLCALCS }
```

Notes

When REMOVECOLCALCS is used, 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.

Related Topics

- [CALCULATE COLUMN](#)
The Essbase Report Writer CALCULATE COLUMN command creates a new report column, performs on-the-fly calculations, and displays the calculation results in the newly-created column.
- [CLEARROWCALC](#)
The Essbase Report Writer CLEARROWCALC command resets the value of the row calculation *name* to #MISSING.
- [CLEARALLROWCALC](#)
The Essbase Report Writer CLEARALLROWCALC command resets the value of all calculated rows to #MISSING.
- [OFFCOLCALCS](#)
The Essbase Report Writer OFFCOLCALCS command disables all column calculations within the report.

- **OFFROWCALCS**
The Essbase Report Writer OFFROWCALCS command temporarily disables all row calculations; for example, those calculations set by CALCULATE ROW.
- **ONCOLCALCS**
The Essbase Report Writer ONCOLCALCS command re-enables column calculations in the report after they have been disabled by OFFCOLCALCS.
- **ONROWCALCS**
The Essbase Report Writer ONROWCALCS command re-enables all row calculations after they have been disabled by OFFROWCALCS. Each subsequent row of data after using the command is calculated.
- **PRINTROW**
The Essbase Report Writer PRINTROW command displays the calculated *rowName* with its current values.
- **SETROWOP**
The Essbase Report Writer SETROWOP command defines on-the-fly calculations for a named row created with CALCULATE ROW.

RENAME

The Essbase Report Writer RENAME command renames a member within the report.

Syntax

```
{ RENAME "newMbrName" } mbrName
```

Parameters

"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

The following report script command renames the Visual member to "Video" in the report.

```
{RENAME "Video"} Visual
```


REPALIAS

The Essbase Report Writer REPALIAS command 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

- Use `<REPALIAS ""` to display alias names for members in all the 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, OUTALTNAMES, OR OUTMBRNAMES.

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

Related Topics

- [OUTALTSELECT](#)
The Essbase Report Writer OUTALTSELECT command selects an alias table in a report script.
- [OUTPUTMEMBERKEY](#)
The Essbase Report Writer OUTPUTMEMBERKEY command displays a member identifier (in addition to the member or alias name) for any duplicate member names. OUTPUTMEMBERKEY applies to duplicate member outlines only.
- [REPALIASMBR](#)
The Essbase Report Writer REPALIASMBR command displays alias names followed by member names for members of the dimension specified in the report output.
- [REPMBR](#)
The Essbase Report Writer REPMBR command displays member names only for members of the dimension specified.
- [REPMBRALIAS](#)
The Essbase Report Writer REPMBRALIAS command displays member names followed by aliases for members of the dimension specified.
- [REPQUALMBR](#)
For the Essbase dimension specified, the Report Writer REPQUALMBR command displays member names for any unique member names, and displays a system generated identifier (a qualified name) for any duplicate member names. REPQUALMBR applies to duplicate member outlines only.

REPALIASMBR

The Essbase Report Writer REPALIASMBR command 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

- Use `<REPALIASMBR ""` to display alias names followed by member names for members of 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, OUTALTNames, OR OUTMBRNames.

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
```

Related Topics

- [OUTALTSELECT](#)
The Essbase Report Writer OUTALTSELECT command selects an alias table in a report script.
- [OUTPUTMEMBERKEY](#)
The Essbase Report Writer OUTPUTMEMBERKEY command displays a member identifier (in addition to the member or alias name) for any duplicate member names. OUTPUTMEMBERKEY applies to duplicate member outlines only.

- **REPALIAS**
The Essbase Report Writer REPALIAS command displays alias names for members of the dimension specified.
- **REPMBR**
The Essbase Report Writer REPMBR command displays member names only for members of the dimension specified.
- **REPMBRALIAS**
The Essbase Report Writer REPMBRALIAS command displays member names followed by aliases for members of the dimension specified.
- **REPQUALMBR**
For the Essbase dimension specified, the Report Writer REPQUALMBR command displays member names for any unique member names, and displays a system generated identifier (a qualified name) for any duplicate member names. REPQUALMBR applies to duplicate member outlines only.

REPMBR

The Essbase Report Writer REPMBR command displays member names only for members of the dimension specified.

Used with the commands [REPALIAS](#), [REPMBRALIAS](#), and [REPALIASMBR](#).

Syntax

```
<REPMBR dimensionname
```

Notes

- Use `<REPMBR ""` to display member names 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, OUTALT NAMES, OR OUTMBR NAMES.

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

```

Related Topics

- [OUTPUTMEMBERKEY](#)
The Essbase Report Writer OUTPUTMEMBERKEY command displays a member identifier (in addition to the member or alias name) for any duplicate member names. OUTPUTMEMBERKEY applies to duplicate member outlines only.
- [REPALIAS](#)
The Essbase Report Writer REPALIAS command displays alias names for members of the dimension specified.
- [REPALIASMBR](#)
The Essbase Report Writer REPALIASMBR command displays alias names followed by member names for members of the dimension specified in the report output.
- [REPMBRALIAS](#)
The Essbase Report Writer REPMBRALIAS command displays member names followed by aliases for members of the dimension specified.
- [REPQUALMBR](#)
For the Essbase dimension specified, the Report Writer REPQUALMBR command displays member names for any unique member names, and displays a system generated identifier (a qualified name) for any duplicate member names. REPQUALMBR applies to duplicate member outlines only.

REPMBRALIAS

The Essbase Report Writer REPMBRALIAS command 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

- Use `<REPMBRALIAS ""` to display member names followed by aliases for members of 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

Related Topics

- [OUTALTSELECT](#)
The Essbase Report Writer OUTALTSELECT command selects an alias table in a report script.
- [OUTPUTMEMBERKEY](#)
The Essbase Report Writer OUTPUTMEMBERKEY command displays a member identifier (in addition to the member or alias name) for any duplicate member names. OUTPUTMEMBERKEY applies to duplicate member outlines only.
- [REPALIAS](#)
The Essbase Report Writer REPALIAS command displays alias names for members of the dimension specified.
- [REPALIASMBR](#)
The Essbase Report Writer REPALIASMBR command displays alias names followed by member names for members of the dimension specified in the report output.
- [REPMBR](#)
The Essbase Report Writer REPMBR command displays member names only for members of the dimension specified.
- [REPQUALMBR](#)
For the Essbase dimension specified, the Report Writer REPQUALMBR command displays member names for any unique member names, and displays a system generated identifier (a qualified name) for any duplicate member names. REPQUALMBR applies to duplicate member outlines only.

REPQUALMBR

For the Essbase dimension specified, the Report Writer REPQUALMBR command displays member names for any unique member names, and displays a system generated identifier (a qualified name) for any duplicate member names. REPQUALMBR applies to duplicate member outlines only.

Syntax

```
<REPQUALMBR dimensionname
```

Notes

- Use <REPQUALMBR "" to apply the command to members in all dimensions.
- Some formatting commands (for example, [RENAME](#)) do not work with REPQUALMBR.

- REQUALMBR cannot be used in combination with the existing commands OUTMBRALT, OUTALTMBR, OUTALT, OUTALTNAMES, OR OUTMBRNames.

Related Topics

- [OUTPUTMEMBERKEY](#)
The Essbase Report Writer OUTPUTMEMBERKEY command displays a member identifier (in addition to the member or alias name) for any duplicate member names. OUTPUTMEMBERKEY applies to duplicate member outlines only.
- [REPALIAS](#)
The Essbase Report Writer REPALIAS command displays alias names for members of the dimension specified.
- [REPALIASMBR](#)
The Essbase Report Writer REPALIASMBR command displays alias names followed by member names for members of the dimension specified in the report output.
- [REPMBR](#)
The Essbase Report Writer REPMBR command displays member names only for members of the dimension specified.
- [REPMBRALIAS](#)
The Essbase Report Writer REPMBRALIAS command displays member names followed by aliases for members of the dimension specified.

RESTRICT

The Essbase Report Writer 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>})
```

Parameters

<column >

@DATACOLUMN (<colNumber>) | @DATACOLUMN (<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 <DESCENDANTS).

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)
-

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 (@DATACOLUMN(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

Related Topics

- [TOP](#)
The TOP Report Writer command in Essbase returns rows with the highest values of a specified data column.
- [BOTTOM](#)
The Essbase Report Writer BOTTOM command returns rows with the lowest values of a specified data column.

- **ORDERBY**
The Essbase Report Writer ORDERBY command orders the rows in a report according to data values in the specified columns.

ROW

The Essbase Report Writer ROW command determines the row dimensions for a report whose member names appear in the data rows of the report.

The member(s) in the ROW command determine which dimensions are displayed in the rows of the report.

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 )
```

Parameters

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.

Related Topics

- **COLUMN**
The Essbase Report Writer COLUMN command defines the dimensions displayed as column members. Column members are displayed above data columns.
- **PAGE**
The Essbase Report Writer PAGE command defines which dimensions are displayed as page members in the final report.

ROWREPEAT

The Essbase Report Writer ROWREPEAT command 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

The default behavior 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

Related Topics

- [NOROWREPEAT](#)
The Essbase Report Writer NOROWREPEAT 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. This is the default.
- [ROW](#)
The Essbase Report Writer ROW command determines the row dimensions for a report whose member names appear in the data rows of the report.

SAVEANDOUTPUT

The SAVEANDOUTPUT command in Essbase Report Writer adds a row member to the report, and creates a new calculated row whose default name is that row member. Its name can be changed using an optional row calculation name.

SAVEANDOUTPUT 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 !
```

Parameters

"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
```

Related Topics

- [CALCULATE COLUMN](#)
The Essbase Report Writer CALCULATE COLUMN command creates a new report column, performs on-the-fly calculations, and displays the calculation results in the newly-created column.
- [CALCULATE ROW](#)
The Essbase Report Writer CALCULATE ROW command 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.
- [CLEARROWCALC](#)
The Essbase Report Writer CLEARROWCALC command resets the value of the row calculation *name* to #MISSING.
- [CLEARALLROWCALC](#)
The Essbase Report Writer CLEARALLROWCALC command resets the value of all calculated rows to #MISSING.
- [OFFCOLCALCS](#)
The Essbase Report Writer OFFCOLCALCS command disables all column calculations within the report.
- [OFFROWCALCS](#)
The Essbase Report Writer OFFROWCALCS command temporarily disables all row calculations; for example, those calculations set by CALCULATE ROW.

- **ONCOLCALCS**
The Essbase Report Writer ONCOLCALCS command re-enables column calculations in the report after they have been disabled by OFFCOLCALCS.
- **ONROWCALCS**
The Essbase Report Writer ONROWCALCS command re-enables all row calculations after they have been disabled by OFFROWCALCS. Each subsequent row of data after using the command is calculated.
- **OUTPUT**
The Essbase Report Writer OUTPUT command resumes output, reversing the action of SUPOUTPUT.
- **PRINTROW**
The Essbase Report Writer PRINTROW command displays the calculated *rowName* with its current values.
- **REMOVECOLCALCS**
The Essbase Report Writer REMOVECOLCALCS command removes all column calculation definitions from the report.
- **SAVEANDOUTPUT**
The SAVEANDOUTPUT command in Essbase Report Writer adds a row member to the report, and creates a new calculated row whose default name is that row member. Its name can be changed using an optional row calculation name.
- **SAVEROW**
The Essbase Report Writer SAVEROW command creates a new calculated row whose default name is specified by *rowMbr*, but which may be renamed with an optional name enclosed in quotation marks.
- **SUPOUTPUT**
The Essbase Report Writer SUPOUTPUT command suppresses all output, except columns, while continuing to process other operations such as calculations or format settings. Use the OUTPUT command to resume output.

SAVEROW

The Essbase Report Writer SAVEROW command creates a new calculated row whose default name is specified by *rowMbr*, but which may be renamed with an optional name enclosed in quotation marks.

SAVEROW 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 !
```

Parameters

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. *rowMbr* 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 report script 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
      !
```

The above report script 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
```


Related Topics

- [SAVEANDOUTPUT](#)
The SAVEANDOUTPUT command in Essbase Report Writer adds a row member to the report, and creates a new calculated row whose default name is that row member. Its name can be changed using an optional row calculation name.

SCALE

The SCALE command in Essbase Report Writer scales the data in the report by multiplying it by a numeric value.

Syntax

```
{ SCALE factor [ columnList ] }
```

Parameters**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

SCALE command affects only the report columns specified in the command, or all columns, if none are specified. Stored data is not affected.

Example

The following report script is designed for the Demo Basic cube, available in the gallery. 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
  !
```

The above report script 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

Related Topics

- [BRACKETS](#)
The Essbase Report Writer BRACKETS command displays parentheses around negative numbers instead of negative signs.
- [COMMAS](#)
The Essbase Report Writer COMMAS command displays commas for numbers greater than 999 after commas have been suppressed with either a SUPCOMMAS or SUPALL command.
- [DECIMAL](#)
The Essbase Report Writer DECIMAL command determines the number of decimal places to display in the report.
- [SUPBRACKETS](#)
The Essbase Report Writer SUPBRACKETS command suppresses the display of parentheses around negative numbers.
- [SUPCOMMAS](#)
The SUPCOMMAS command in Essbase Report Writer suppresses the display of commas in numbers greater than 999. The default behavior is to display the commas.

SETCENTER

The Essbase Report Writer SETCENTER command sets a new centerline position on the page.

Syntax

```
{ SETCENTER charPosition }
```

Parameters**charPosition**

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

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

The Essbase Report Writer SETROWOP command defines on-the-fly calculations for a named row created with CALCULATE ROW.

SETROWOP determines the calculation for the calculated row specified in *rowCalcName*. The following table lists the operators you can use for the *operation* in the command:

Table 4-4 Operators for CALCULATE ROW

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 ] }
```

Parameters

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.

Related Topics

- [CALCULATE ROW](#)

The Essbase Report Writer CALCULATE ROW command 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.

- **CLEARROWCALC**
The Essbase Report Writer CLEARROWCALC command resets the value of the row calculation *name* to #MISSING.
- **CLEARALLROWCALC**
The Essbase Report Writer CLEARALLROWCALC command resets the value of all calculated rows to #MISSING.
- **OFFCOLCALCS**
The Essbase Report Writer OFFCOLCALCS command disables all column calculations within the report.
- **OFFROWCALCS**
The Essbase Report Writer OFFROWCALCS command temporarily disables all row calculations; for example, those calculations set by CALCULATE ROW.
- **ONCOLCALCS**
The Essbase Report Writer ONCOLCALCS command re-enables column calculations in the report after they have been disabled by OFFCOLCALCS.
- **ONROWCALCS**
The Essbase Report Writer ONROWCALCS command re-enables all row calculations after they have been disabled by OFFROWCALCS. Each subsequent row of data after using the command is calculated.
- **OUTPUT**
The Essbase Report Writer OUTPUT command resumes output, reversing the action of SUPOUTPUT.
- **PRINTROW**
The Essbase Report Writer PRINTROW command displays the calculated *rowName* with its current values.
- **REMOVECOLCALCS**
The Essbase Report Writer REMOVECOLCALCS command removes all column calculation definitions from the report.
- **SAVEANDOUTPUT**
The SAVEANDOUTPUT command in Essbase Report Writer adds a row member to the report, and creates a new calculated row whose default name is that row member. Its name can be changed using an optional row calculation name.
- **SAVEROW**
The Essbase Report Writer SAVEROW command creates a new calculated row whose default name is specified by *rowMbr*, but which may be renamed with an optional name enclosed in quotation marks.
- **SUPOUTPUT**
The Essbase Report Writer SUPOUTPUT command suppresses all output, except columns, while continuing to process other operations such as calculations or format settings. Use the OUTPUT command to resume output.

SINGLECOLUMN

SINGLECOLUMN is an Essbase Report Writer formatting command that displays a column heading when there is only one column member extracted in the report.

Syntax

```
<SINGLECOLUMN
```

Example

The following report script is designed for the Sample Basic cube, available in the gallery.

```
<SINGLECOLUMN
{SUPPAGEHEAD}
<COLUMN (Year)
<ROW (Measures)
Profit Inventory Ratios
Qtr1
!
```

The above report script produces the following report:

	Qtr1
	=====
Profit	24,703
Inventory	117,405
Ratios	55

Related Topics

- [COLHEADING](#)
The Essbase Report Writer COLHEADING command 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.
- [PAGEHEADING](#)
The Essbase Report Writer PAGEHEADING command displays the page heading before the next data-output row.
- [SUPCOLHEADING](#)
The SUPCOLHEADING command in Essbase Report Writer suppresses display of default column headings.
- [SUPPAGEHEADING](#)
The Essbase Report Writer SUPPAGEHEADING command suppresses display of the page member heading whenever a heading is generated.

SKIP

The Essbase Report Writer SKIP command outputs a specified number of blank lines in the report. If the number is not specified, it outputs a single line. The default value is single skip.

Syntax

```
{SKIP n }
```

Parameters

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

The following report script is designed for the Sample Basic cube, available in the gallery.

```
<PAGE (Measures, Market)
Texas Sales
    <COLUMN (Scenario, Year)
        Actual Budget
        Jan Feb
<ROW (Market)
<DESCENDANTS "100"
{SKIP 2}
<DESCENDANTS "200"
<DESCENDANTS "300"
    !
```

The above report script inserts two blank lines between the rows containing descendants of member 100 and descendants of members 200 and 300.

Related Topics

- [NEWPAGE](#)
The Essbase Report Writer NEWPAGE command inserts a new page in the report regardless of how many lines have been generated for the current page.
- [NOSKIPONDIMENSION](#)
The Essbase Report Writer NOSKIPONDIMENSION command prevents insertion of a new line when a member from the same dimension as the input member changes in a row of the report.
- [SKIPONDIMENSION](#)
The Essbase Report Writer SKIPONDIMENSION command inserts a blank line when a member from the same dimension as the specified member changes on the next line in the report.

SKIPONDIMENSION

The Essbase Report Writer SKIPONDIMENSION command 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 }
```

Parameters

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

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. 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 following report script is designed for the Demo Basic cube, available in the gallery. The command {SKIPONDIMENSION Year} 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
  !
```

The above report script 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

Related Topics

- [NOPAGEONDIMENSION](#)
The Essbase Report Writer NOPAGEONDIMENSION 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.
- [NOSKIPONDIMENSION](#)
The Essbase Report Writer NOSKIPONDIMENSION command prevents insertion of a new line when a member from the same dimension as the input member changes in a row of the report.
- [PAGEONDIMENSION](#)
The Essbase Report Writer PAGEONDIMENSION command 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.

SORTALTNAMES

The SORTALTNAMES Report Writer command for Essbase alphabetically sorts members by their alternate names within a member selection command (for example, <CHILDREN).

Syntax

```
<SORTALTNAMES
```

Notes

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.

SORTALTNAMES 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 report script is designed for the Demo Basic cube, available in the gallery section of the Essbase file catalog.

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.

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual
  <COLUMN (Year)
  <CHILDREN Year
<ROW (Product)
<IDESCENDANTS Product
{NOINDENTGEN}
  !
```

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual
{OUTALT NAMES}
  <COLUMN (Year)
  <CHILDREN Year
<ROW (Product)
<SORTALT NAMES
{NOINDENTGEN}
<IDESCENDANTS Product
  !
```

The above report script produces the following reports:

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

Chicago Sales Actual

	Q1	Q2	Q3	Q4
	=====	=====	=====	=====
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

Related Topics

- [ALLINSAMEDIM](#)
The Essbase Report Writer ALLINSAMEDIM command selects all the members from the same dimension as the specified dimension member for the report.
- [CHILDREN](#)
The Essbase Report Writer CHILDREN command selects all members in the level immediately below the specified member, excluding the specified member.
- [DESCENDANTS](#)
The Essbase Report Writer DESCENDANTS command adds the descendants of *mbrName* to the report, excluding *mbrName*.
- [SORTASC](#)
The SORTASC Report Writer command for Essbase specifies an ascending sort order to be used for member commands.
- [SORTDESC](#)
The SORTDESC Report Writer command for Essbase specifies a descending, hierarchical sort order to be used for member commands.
- [SORTGEN](#)
The SORTGEN command in Report Writer sorts members added with a member command, such as <CHILDREN, according to the generation of the member in the Essbase outline. The top member of the dimension is generation 1, its children are generation 2, etc.
- [SORTLEVEL](#)
The SORTLEVEL command in Essbase Report Writer sorts all members added with a member selection command, such as <CHILDREN, according to the level of the member.
- [SORTMBRNames](#)
The SORTMBRNames command in Essbase Report Writer sorts members added with a member selection command, such as <CHILDREN, alphabetically by member name. Not affected are 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.
- [SORTNONE](#)
The Essbase Report Writer SORTNONE command disables all previous sorting commands.

SORTASC

The SORTASC Report Writer command for Essbase specifies an ascending sort order to be used for member commands.

Syntax

```
<SORTASC
```

Notes

This command determines the order in which members are sorted in member commands in the report specification. 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.

SORTASC 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.

SORTASC can be used to restore the default (ascending) sort order. It reverses the effects of a previously-specified SORTDESC command.

Related Topics

- [ALLINSAMEDIM](#)
The Essbase Report Writer ALLINSAMEDIM command selects all the members from the same dimension as the specified dimension member for the report.
- [CHILDREN](#)
The Essbase Report Writer CHILDREN command selects all members in the level immediately below the specified member, excluding the specified member.
- [DESCENDANTS](#)
The Essbase Report Writer DESCENDANTS command adds the descendants of *mbrName* to the report, excluding *mbrName*.
- [SORTALTNAMES](#)
The SORTALTNAMES Report Writer command for Essbase alphabetically sorts members by their alternate names within a member selection command (for example, <CHILDREN).
- [SORTDESC](#)
The SORTDESC Report Writer command for Essbase specifies a descending, hierarchical sort order to be used for member commands.
- [SORTGEN](#)
The SORTGEN command in Report Writer sorts members added with a member command, such as <CHILDREN, according to the generation of the member in the Essbase outline. The top member of the dimension is generation 1, its children are generation 2, etc.
- [SORTLEVEL](#)
The SORTLEVEL command in Essbase Report Writer sorts all members added with a member selection command, such as <CHILDREN, according to the level of the member.
- [SORTMBRNames](#)
The SORTMBRNames command in Essbase Report Writer sorts members added with a member selection command, such as <CHILDREN, alphabetically by member name. Not affected are 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.
- [SORTNONE](#)
The Essbase Report Writer SORTNONE command disables all previous sorting commands.

SORTDESC

The SORTDESC Report Writer command for Essbase specifies a descending, hierarchical sort order to be used for member commands.

Syntax

```
<SORTDESC
```

Notes

SORTDESC determines the order in which items are sorted in member commands in the report specification. 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.

SORTDESC 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 report script example is based on Sample Basic.

```
<PAGE (Market, Measures)
Massachusetts Sales
<COLUMN (Scenario, Year)
Actual Budget
Jan Feb Mar
<ROW (Product)
<SORTDESC
<CHILDREN Product
!
```

The above example produces the following report:

	Massachusetts Sales					
	Actual			Budget		
	Jan	Feb	Mar	Jan	Feb	Mar
Product	1,251	1,206	1,203	1,170	1,130	1,120
Diet	#Missing	#Missing	#Missing	#Missing	#Missing	#Missing
400	160	136	132	160	140	130
300	130	132	129	100	100	100
200	467	468	450	450	450	430
100	494	470	492	460	440	460

Related Topics

- [ALLINSAMEDIM](#)
The Essbase Report Writer ALLINSAMEDIM command selects all the members from the same dimension as the specified dimension member for the report.
- [DESCENDANTS](#)
The Essbase Report Writer DESCENDANTS command adds the descendants of *mbrName* to the report, excluding *mbrName*.
- [SORTASC](#)
The SORTASC Report Writer command for Essbase specifies an ascending sort order to be used for member commands.
- [SORTALTNAMES](#)
The SORTALTNAMES Report Writer command for Essbase alphabetically sorts members by their alternate names within a member selection command (for example, <CHILDREN).
- [SORTGEN](#)
The SORTGEN command in Report Writer sorts members added with a member command, such as <CHILDREN, according to the generation of the member in the Essbase outline. The top member of the dimension is generation 1, its children are generation 2, etc.
- [SORTLEVEL](#)
The SORTLEVEL command in Essbase Report Writer sorts all members added with a member selection command, such as <CHILDREN, according to the level of the member.
- [SORTMBRNAMES](#)
The SORTMBRNAMES command in Essbase Report Writer sorts members added with a member selection command, such as <CHILDREN, alphabetically by member name. Not affected are 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.
- [SORTNONE](#)
The Essbase Report Writer SORTNONE command disables all previous sorting commands.

SORTGEN

The SORTGEN command in Report Writer sorts members added with a member command, such as <CHILDREN, according to the generation of the member in the Essbase outline. The top member of the dimension is generation 1, its children are generation 2, etc.

The following are not affected by this command:

- Members entered directly in the report specification without using a member selection command.
- Calculated rows and column names.
- Member commands encountered in the script prior to the SORTGEN command.

SORTGEN must precede the member selection commands; for example, CHILDREN or DESCENDANTS. If no sorting commands are used, members are output in hierarchical order based on the 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 report script example is based on Sample Basic.

```
<PAGE (Product, Measures)
East Sales
<COLUMN (Scenario, Year)

Actual Budget
Jan Feb Mar
<ROW (Market)
<SORTGEN
<IDESCENDANTS Market
    !
```

The example above 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

Related Topics

- [ALLINSAMEDIM](#)
The Essbase Report Writer ALLINSAMEDIM command selects all the members from the same dimension as the specified dimension member for the report.
- [CHILDREN](#)
The Essbase Report Writer CHILDREN command selects all members in the level immediately below the specified member, excluding the specified member.
- [DESCENDANTS](#)
The Essbase Report Writer DESCENDANTS command adds the descendants of *mbrName* to the report, excluding *mbrName*.
- [SORTASC](#)
The SORTASC Report Writer command for Essbase specifies an ascending sort order to be used for member commands.
- [SORTALTNAMES](#)
The SORTALTNAMES Report Writer command for Essbase alphabetically sorts members by their alternate names within a member selection command (for example, <CHILDREN).
- [SORTDESC](#)
The SORTDESC Report Writer command for Essbase specifies a descending, hierarchical sort order to be used for member commands.
- [SORTLEVEL](#)
The SORTLEVEL command in Essbase Report Writer sorts all members added with a member selection command, such as <CHILDREN, according to the level of the member.
- [SORTMBRNames](#)
The SORTMBRNames command in Essbase Report Writer sorts members added with a member selection command, such as <CHILDREN, alphabetically by member name. Not affected are 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.
- [SORTNONE](#)
The Essbase Report Writer SORTNONE command disables all previous sorting commands.

SORTLEVEL

The SORTLEVEL command in Essbase Report Writer 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.

If used, SORTLEVEL 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 report script example is based on Sample Basic.

```
<PAGE (Product, Measures)
East Sales
<COLUMN (Scenario, Year)

Actual Budget
Jan Feb Mar
<ROW (Market)
<SORTLEVEL
<IDESCENDANTS Market
!
```

The above 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

Related Topics

- [ALLINSAMEDIM](#)
The Essbase Report Writer ALLINSAMEDIM command selects all the members from the same dimension as the specified dimension member for the report.
- [CHILDREN](#)
The Essbase Report Writer CHILDREN command selects all members in the level immediately below the specified member, excluding the specified member.
- [DESCENDANTS](#)
The Essbase Report Writer DESCENDANTS command adds the descendants of *mbrName* to the report, excluding *mbrName*.
- [SORTASC](#)
The SORTASC Report Writer command for Essbase specifies an ascending sort order to be used for member commands.
- [SORTALTNAMES](#)
The SORTALTNAMES Report Writer command for Essbase alphabetically sorts members by their alternate names within a member selection command (for example, <CHILDREN).
- [SORTDESC](#)
The SORTDESC Report Writer command for Essbase specifies a descending, hierarchical sort order to be used for member commands.
- [SORTGEN](#)
The SORTGEN command in Report Writer sorts members added with a member command, such as <CHILDREN, according to the generation of the member in the Essbase outline. The top member of the dimension is generation 1, its children are generation 2, etc.
- [SORTMBRNames](#)
The SORTMBRNames command in Essbase Report Writer sorts members added with a member selection command, such as <CHILDREN, alphabetically by member name. Not affected are 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.
- [SORTNONE](#)
The Essbase Report Writer SORTNONE command disables all previous sorting commands.

SORTMBRNames

The SORTMBRNames command in Essbase Report Writer sorts members added with a member selection command, such as <CHILDREN, alphabetically by member name. Not affected are 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.

If used, SORTMBRNames 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 report script example is based on Sample Basic.

```
<PAGE (Product, Measures)
Sales
<COLUMN (Scenario, Year)
Actual Budget
Jan Feb Mar
<ROW (Market)
<SORTMBRNames
<IDESCENDANTS South
    !
```

The above example produces the following report:

Product Sales						
	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

The Essbase Report Writer SORTNONE command disables all previous sorting commands.

Syntax

```
<SORTNONE
```

Notes

This command disables all previous member sorting commands, so that members added to the report with member selection commands are added in outline order.

Related Topics

- [ALLINSAMEDIM](#)
The Essbase Report Writer ALLINSAMEDIM command selects all the members from the same dimension as the specified dimension member for the report.
- [DESCENDANTS](#)
The Essbase Report Writer DESCENDANTS command adds the descendants of *mbrName* to the report, excluding *mbrName*.
- [SORTALTNAMES](#)
The SORTALTNAMES Report Writer command for Essbase alphabetically sorts members by their alternate names within a member selection command (for example, <CHILDREN).
- [SORTDESC](#)
The SORTDESC Report Writer command for Essbase specifies a descending, hierarchical sort order to be used for member commands.
- [SORTGEN](#)
The SORTGEN command in Report Writer sorts members added with a member command, such as <CHILDREN, according to the generation of the member in the Essbase outline. The top member of the dimension is generation 1, its children are generation 2, etc.
- [SORTLEVEL](#)
The SORTLEVEL command in Essbase Report Writer sorts all members added with a member selection command, such as <CHILDREN, according to the level of the member.
- [SORTMBRNames](#)
The SORTMBRNames command in Essbase Report Writer sorts members added with a member selection command, such as <CHILDREN, alphabetically by member name. Not affected are 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.

SPARSE

The SPARSE command in Report Writer 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.

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 cube. 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 cube 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 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.

Related Topics

- [SUPMISSINGROWS](#)
The Essbase Report Writer SUPMISSINGROWS command can control the display of empty values that are #MISSING in the report. It suppresses the display of rows that contain only #MISSING values.

STARTHEADING

The Essbase Report Writer STARTHEADING 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.

Syntax

```
{ STARTHEADING }
```

Notes

- 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. 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.
- 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.
- To use a substitution variable in a heading, you must use the TEXT command. Example:

```
{STARTHEADING TEXT 2 "Prepared by:" 14 "*USERNAME"  
  C "The Electronics Club" 60 "*PAGESTRING"  
  TEXT C "Quarterly Sales by City" 60 "*DATE"  
  SUPPAGEHEADING  
  Text 2 &Month  
  TEXT 2 "*PAGEHDR" SKIP ENDHEADING}  
!
```

 **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

If you use STARTHEADING, it replaces the 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
  !
```

The above report script example produces the following report:

```
Prepared by:ksmith      The Electronics Club      Page: 1
                       Quarterly Sales by City    06/10/20
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

Related Topics

- [ENDHEADING](#)
The Essbase Report Writer ENDHEADING command ends the definition of the custom page heading displayed at the top of each page.
- [HEADING](#)
The Essbase Report Writer HEADING command displays the page heading: either the default heading, or the heading as defined with the STARTHEADING and ENDHEADING commands.
- [IMMHEADING](#)
The Essbase Report Writer IMMHEADING command forces the immediate display of the heading without waiting for the next non-suppressed data row.
- [SUPCOLHEADING](#)
The SUPCOLHEADING command in Essbase Report Writer suppresses display of default column headings.
- [SUPHEADING](#)
The Essbase Report Writer SUPHEADING command suppresses the display of the default heading (page header and column headers) or custom header, if defined, at the top of each page.
- [SUPPAGEHEADING](#)
The Essbase Report Writer SUPPAGEHEADING command suppresses display of the page member heading whenever a heading is generated.

SUDA

The Essbase Report Writer SUDA command selects members based on a common attribute, defined as a UDA (user-defined attribute), along with their shared counterparts.

Syntax

```
<SUDA (dimName, udaStr)
```

Parameters

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 prototype 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 prototype 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))
```

Related Topics

- [UDA](#)
The UDA Report Writer command in Essbase selects and reports on members based on a common attribute, defined as a UDA (user-defined attribute).

SUPALL

The SUPALL command in Essbase Report Writer suppresses the display of the page and column headings, all member names, page breaks, commas, and brackets.

Syntax

```
{ SUPALL }
```

Notes

With SUPALL, you display the data of the report, and any text displayed as the result of the TEXT command. SUPALL command is equivalent to SUPHEADING, SUPPAGEHEADING,

SUPCOLHEADING, SUPNAMES, SUPBRACKETS, SUPFEED, and SUPCOMMAS, combined.

Example

The following report script is designed for the Demo Basic cube, available in the gallery section of the Essbase file catalog.

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual
    <COLUMN (Year)
        <CHILDREN Year
<ROW (Product)
<ICHILDREN Audio
    !

{ SUPALL }
Boston Sales Actual
    <CHILDREN Year
<ICHILDREN Audio
    !
```

The above 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	

Related Topics

- [SUPBRACKETS](#)
The Essbase Report Writer SUPBRACKETS command suppresses the display of parentheses around negative numbers.
- [SUPCOLHEADING](#)
The SUPCOLHEADING command in Essbase Report Writer suppresses display of default column headings.

- **SUPCOMMAS**
The SUPCOMMAS command in Essbase Report Writer suppresses the display of commas in numbers greater than 999. The default behavior is to display the commas.
- **SUPCURHEADING**
The Essbase Report Writer SUPCURHEADING command suppresses the display of currency information when you use the CURRENCY command to convert the data values in your report to a specified currency.
- **SUPEMPTYROWS**
The Essbase Report Writer SUPEMPTYROWS command suppresses the display of empty rows – that is, rows that have only 0 or #MISSING values in the row.
- **SUPEUROPEAN**
The Essbase Report Writer SUPEUROPEAN command disables the European method for displaying numbers.
- **SUPFEED**
The Essbase Report Writer SUPFEED command suppresses the automatic insertion of a physical page break whenever the number of lines on a page exceeds the current PAGELENGTH setting.
- **SUPHEADING**
The Essbase Report Writer SUPHEADING command suppresses the display of the default heading (page header and column headers) or custom header, if defined, at the top of each page.
- **SUPMISSINGROWS**
The Essbase Report Writer SUPMISSINGROWS command can control the display of empty values that are #MISSING in the report. It suppresses the display of rows that contain only #MISSING values.
- **SUPNAMES**
The Essbase Report Writer SUPNAMES command suppresses the display of row member names in the final report.
- **SUPPAGEHEADING**
The Essbase Report Writer SUPPAGEHEADING command suppresses display of the page member heading whenever a heading is generated.
- **SUPZEROROWS**
The Essbase Report Writer SUPZEROROWS command is one of the commands that helps deal with empty values in a report. It suppresses the display of rows that have only 0 values.

SUPBRACKETS

The Essbase Report Writer SUPBRACKETS command suppresses the display of parentheses around negative numbers.

Syntax

```
{ SUPBRACKETS }
```

Notes

When you use SUPBRACKETS, the negative sign,(-), rather than parentheses, indicates negative numbers.

Example

```
{SUPBRACKETS}
```

displays (34.43) as -34.43.

Related Topics

- [COMMAS](#)
The Essbase Report Writer COMMAS command displays commas for numbers greater than 999 after commas have been suppressed with either a SUPCOMMAS or SUPALL command.
- [DECIMAL](#)
The Essbase Report Writer DECIMAL command determines the number of decimal places to display in the report.
- [SUPALL](#)
The SUPALL command in Essbase Report Writer suppresses the display of the page and column headings, all member names, page breaks, commas, and brackets.
- [SUPBRACKETS](#)
The Essbase Report Writer SUPBRACKETS command suppresses the display of parentheses around negative numbers.
- [SUPCOMMAS](#)
The SUPCOMMAS command in Essbase Report Writer suppresses the display of commas in numbers greater than 999. The default behavior is to display the commas.

SUPCOLHEADING

The SUPCOLHEADING command in Essbase Report Writer suppresses display of default column headings.

Syntax

```
{ SUPCOLHEADING }
```

Notes

Unless a custom heading is defined, Report Writer displays 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

The following report script is designed for the Demo Basic cube, available in the gallery section of the Essbase file catalog.

```
<PAGE (Market, Accounts, Scenario)
Boston Sales Actual
    <COLUMN (Year)
    <CHILDREN Year
<ROW (Product)
<ICHILDREN Audio
!
```

```
{ SUPCOLHEADING }
Boston Sales Actual
    <COLUMN (Year)
    <CHILDREN Year
<ROW (Product)
<ICHILDREN Audio
    !
```

The above example produces the following reports:

Boston Sales Actual				
	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

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

Related Topics

- [COLHEADING](#)
The Essbase Report Writer COLHEADING command 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.
- [NAMESON](#)
The Essbase Report Writer NAMESON command turns on the display of column(s) of row member names.
- [PAGEHEADING](#)
The Essbase Report Writer PAGEHEADING command displays the page heading before the next data-output row.
- [SUPNAMES](#)
The Essbase Report Writer SUPNAMES command suppresses the display of row member names in the final report.
- [SUPPAGEHEADING](#)
The Essbase Report Writer SUPPAGEHEADING command suppresses display of the page member heading whenever a heading is generated.

SUPCOMMAS

The SUPCOMMAS command in Essbase Report Writer suppresses the display of commas in numbers greater than 999. The default behavior is to display the commas.

Syntax

```
{ SUPCOMMAS }
```

Example

The following report script command displays the number 12,234,534.23 as 12234534.23.

```
{ SUPCOMMAS }
```

Related Topics

- [BRACKETS](#)
The Essbase Report Writer BRACKETS command displays parentheses around negative numbers instead of negative signs.
- [COMMAS](#)
The Essbase Report Writer COMMAS command displays commas for numbers greater than 999 after commas have been suppressed with either a SUPCOMMAS or SUPALL command.
- [DECIMAL](#)
The Essbase Report Writer DECIMAL command determines the number of decimal places to display in the report.
- [SUPBRACKETS](#)
The Essbase Report Writer SUPBRACKETS command suppresses the display of parentheses around negative numbers.

SUPCURHEADING

The Essbase Report Writer SUPCURHEADING command 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.

Related Topics

- [CURHEADING](#)
The Essbase Report Writer CURHEADING command enables the display of the currency conversion heading.

- **CURRENCY**
The Essbase Report Writer CURRENCY command 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 cube: only in the report.

SUPEMPTYROWS

The Essbase Report Writer SUPEMPTYROWS command suppresses the display of empty rows – that is, rows that have only 0 or #MISSING values in the row.

Syntax

```
{ SUPEMPTYROWS }
```

Notes

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
```

Related Topics

- **INCEMPTYROWS**
The Essbase Report Writer INCEMPTYROWS command displays empty rows of data, or rows that contain only zeros or #MISSING data values, in the final report.
- **INCMISSINGROWS**
The Essbase Report Writer INCMISSINGROWS command displays missing rows of data, or rows that contain all #MISSING data values, in the final report.
- **INCZEROROWS**
The Essbase Report Writer INCZEROROWS command includes rows that contain only data values of zero in the final report.
- **SUPMISSINGROWS**
The Essbase Report Writer SUPMISSINGROWS command can control the display of empty values that are #MISSING in the report. It suppresses the display of rows that contain only #MISSING values.
- **SUPZEROROWS**
The Essbase Report Writer SUPZEROROWS command is one of the commands that helps deal with empty values in a report. It suppresses the display of rows that have only 0 values.

SUPEUROPEAN

The Essbase Report Writer SUPEUROPEAN command 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, to reverse its effect.

Default Value

Non-European is the default.

Example

See the example for EUROPEAN.

Related Topics

- [EUROPEAN](#)
The Essbase Report Writer EUROPEAN command enables non-US number formatting by switching commas and decimal points in report data values.

SUPFEED

The Essbase Report Writer SUPFEED command 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

The SUPFEED command disables the FEEDON command. The FEEDON command re-enables physical page breaks. The default page length is 66 lines unless reset with the PAGELENGTH command.

Default Value

Automatic page break insertion is suppressed by default when performing ad-hoc reports in a grid client.

Related Topics

- [FEEDON](#)
The Essbase Report Writer FEEDON command enables page break insertion when the number of output lines on a page is greater than the PAGELENGTH setting.

- **NEWPAGE**
The Essbase Report Writer NEWPAGE command inserts a new page in the report regardless of how many lines have been generated for the current page.
- **PAGELength**
The Essbase Report Writer PAGELength command sets the maximum number of lines for one page in the report.

SUPFORMATS

The Essbase Report Writer SUPFORMATS command suppresses formats that produce extra output, such as underlines and line 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

Formats are set to "ON" by default when the SUPMASK, SUPMISSING, or SUPZERO commands are used.

Related Topics

- **INCFORMATS**
The Essbase Report Writer INCFORMATS command controls the formats affected by the following commands: SUPMASK, SUPMISSING, and SUPZERO.

SUPHEADING

The Essbase Report Writer SUPHEADING command 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 by default.

Example

See the example for STARTHEADING.

Related Topics

- [ENDHEADING](#)
The Essbase Report Writer ENDHEADING command ends the definition of the custom page heading displayed at the top of each page.
- [HEADING](#)
The Essbase Report Writer HEADING command displays the page heading: either the default heading, or the heading as defined with the STARTHEADING and ENDHEADING commands.
- [IMMHEADING](#)
The Essbase Report Writer IMMHEADING command forces the immediate display of the heading without waiting for the next non-suppressed data row.
- [STARTHEADING](#)
The Essbase Report Writer STARTHEADING 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.

SUPMASK

The Essbase Report Writer SUPMASK command 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.

Related Topics

- [MASK](#)
The Essbase Report Writer MASK command overwrites the text in each output row with the specified characters at the specified position.
- [TEXT](#)
The TEXT Report Writer command in Essbase inserts text or other information on a new line in the report.

SUPMISSINGROWS

The Essbase Report Writer SUPMISSINGROWS command can control the display of empty values that are #MISSING in the report. It suppresses the display of rows that contain only #MISSING values.

Syntax

```
{ SUPMISSINGROWS }
```

Example

The following report script is designed for the Sample Basic cube, available in the gallery.

```
<PAGE("Measures")
<COLUMN("Scenario", "Year")
<ROW("Market", "Product")
"Sales"
"Scenario"
"Jan" "Feb" "Mar"
"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"
!
```

```
<PAGE("Measures")
<COLUMN("Scenario", "Year")
<ROW("Market", "Product")
{SUPMISSINGROWS}
{NOINDENTGEN}
"Sales"
"Scenario"
"Jan" "Feb" "Mar"
"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"
!
```

The above report script example produces the following reports. The second report has missing rows suppressed.

		Sales Scenario		
		Jan	Feb	Mar
		=====	=====	=====
New York	Product	2,479	2,625	2,601
	100	678	645	675
	100-10	678	645	675
	100-20	#Missing	#Missing	#Missing
	100-30	#Missing	#Missing	#Missing

200	551	641	586
200-10	61	61	63
200-20	#Missing	#Missing	#Missing
200-30	#Missing	#Missing	#Missing
200-40	490	580	523
300	663	675	695
300-10	483	495	513
300-20	180	180	182
300-30	#Missing	#Missing	#Missing
400	587	664	645
400-10	234	232	234
400-20	219	243	213
400-30	134	189	198
Diet	#Missing	#Missing	#Missing

Sales Scenario

		Jan	Feb	Mar
		=====	=====	=====
New York	Product	2,479	2,625	2,601
	100	678	645	675
	100-10	678	645	675
	200	551	641	586
	200-10	61	61	63
	200-40	490	580	523
	300	663	675	695
	300-10	483	495	513
	300-20	180	180	182
	400	587	664	645
	400-10	234	232	234
	400-20	219	243	213
	400-30	134	189	198

Related Topics

- [INCEMPTYROWS](#)
The Essbase Report Writer INCEMPTYROWS command displays empty rows of data, or rows that contain only zeros or #MISSING data values, in the final report.
- [INCMISSINGROWS](#)
The Essbase Report Writer INCMISSINGROWS command displays missing rows of data, or rows that contain all #MISSING data values, in the final report.
- [INCZEROROWS](#)
The Essbase Report Writer INCZEROROWS command includes rows that contain only data values of zero in the final report.
- [SUPEMPTYROWS](#)
The Essbase Report Writer SUPEMPTYROWS command suppresses the display of empty rows – that is, rows that have only 0 or #MISSING values in the row.
- [SUPZEROROWS](#)
The Essbase Report Writer SUPZEROROWS command is one of the commands that helps deal with empty values in a report. It suppresses the display of rows that have only 0 values.

SUPNAMES

The Essbase Report Writer SUPNAMES command 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 report script example is based on Demo Basic.

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual

    <COLUMN (Year)
    <CHILDREN Year

<ROW (Product)
<CHILDREN Audio
    !

{ SUPNAMES }
Boston Sales Actual
    <CHILDREN Year
<CHILDREN Audio
    !
```

The above 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

Related Topics

- [COLHEADING](#)
The Essbase Report Writer COLHEADING command 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.
- [NAMESON](#)
The Essbase Report Writer NAMESON command turns on the display of column(s) of row member names.
- [PAGEHEADING](#)
The Essbase Report Writer PAGEHEADING command displays the page heading before the next data-output row.
- [SUPCOLHEADING](#)
The SUPCOLHEADING command in Essbase Report Writer suppresses display of default column headings.
- [SUPPAGEHEADING](#)
The Essbase Report Writer SUPPAGEHEADING command suppresses display of the page member heading whenever a heading is generated.

SUOUTPUT

The Essbase Report Writer SUOUTPUT command 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

The following report script is designed for the Demo Basic cube, available in the gallery.

```
<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
<ICCHILDREN Audio
      !
```

This script produces the same report as shown in the [SUPNAMES](#) example.

Related Topics

- [OUTPUT](#)
The Essbase Report Writer OUTPUT command resumes output, reversing the action of SUPOUTPUT.

SUPPAGEHEADING

The Essbase Report Writer SUPPAGEHEADING command suppresses display of the page member heading whenever a heading is generated.

Syntax

```
{ SUPPAGEHEADING }
```

Notes

SUPPAGEHEADING 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

The following report script is designed for the Demo Basic cube, available in the gallery.

```
<PAGE (Market, Accounts, Scenario)
{ STARTHEADING
  SUPPAGEHEADING
  TEXT 2 "*DATETIME" C "Audio Sales Report" 65 "*PAGESTRING" SKIP
  TEXT 2 "City: " 12 "*PAGEHDR 1"
  TEXT 2 "Account: " 12 "*PAGEHDR 2" SKIP
  ENDHEADING }
```

```
Boston Sales Actual
      <CHILDREN Year
<ICCHILDREN Audio
      !
```

The above example produces the following report:

```
06/10/20 07:07:13          Audio Sales Report          Page: 1

City:      Boston
Account:   Sales
```

	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

Related Topics

- [COLHEADING](#)
The Essbase Report Writer COLHEADING command 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.
- [HEADING](#)
The Essbase Report Writer HEADING command displays the page heading: either the default heading, or the heading as defined with the STARTHEADING and ENDHEADING commands.
- [IMMHEADING](#)
The Essbase Report Writer IMMHEADING command forces the immediate display of the heading without waiting for the next non-suppressed data row.
- [NAMESON](#)
The Essbase Report Writer NAMESON command turns on the display of column(s) of row member names.
- [PAGEHEADING](#)
The Essbase Report Writer PAGEHEADING command displays the page heading before the next data-output row.
- [SUPCOLHEADING](#)
The SUPCOLHEADING command in Essbase Report Writer suppresses display of default column headings.
- [SUPNAMES](#)
The Essbase Report Writer SUPNAMES command suppresses the display of row member names in the final report.
- [TEXT](#)
The TEXT Report Writer command in Essbase inserts text or other information on a new line in the report.

SUPSHARE

The Essbase Report Writer SUPSHARE command 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

SUPSHARE 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 is the default behavior – all instances of shared members are displayed.

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 script causes shared members to appear only once in the output report:

```
{SUPMISSINGROWS}
<SUPSHARE
<PAGE (Measures, Market, Scenario)
Sales West Actual
<COLUMN (Year)
<IDESCENDANTS Qtr1
<ROW (Product)
lev0,Product
!
```

As shown in the following report, the shared members appear only once.

	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

Related Topics

- [SUPSHAREOFF](#)
The Essbase Report Writer SUPSHAREOFF command reinstates the display of later instances of shared members after they have been suppressed using the SUPSHARE command.

SUPSHAREOFF

The Essbase Report Writer 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 is the default behavior – all instances of shared members are displayed.

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 script excerpt uses SUPSHAREOFF to reinstate the shared member display so that the shared members appear twice in the report.

```
{SUPMISSINGROWS}
<SUPSHAREOFF
<PAGE (Measures, Market, Scenario)
Sales West Actual
<COLUMN (Year)
<IDESCENDANTS Qtr1
<ROW (Product)
lev0,Product
!
```

The report script excerpt above returns the following data. The example assumes that you have used SUPSHARE in a previous report in the report script.

	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

Related Topics

- [SUPSHARE](#)
The Essbase Report Writer SUPSHARE command suppresses the display of later instances of shared members when you use generation or level names to extract data for your report.

SUPZEROROWS

The Essbase Report Writer SUPZEROROWS command is one of the commands that helps deal with empty values in a report. It 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
```

Related Topics

- [INCEMPTYROWS](#)
The Essbase Report Writer INCEMPTYROWS command displays empty rows of data, or rows that contain only zeros or #MISSING data values, in the final report.
- [INCZEROROWS](#)
The Essbase Report Writer INCZEROROWS command includes rows that contain only data values of zero in the final report.
- [SUPEMPTYROWS](#)
The Essbase Report Writer SUPEMPTYROWS command suppresses the display of empty rows – that is, rows that have only 0 or #MISSING values in the row.
- [SUPMISSINGROWS](#)
The Essbase Report Writer SUPMISSINGROWS command can control the display of empty values that are #MISSING in the report. It suppresses the display of rows that contain only #MISSING values.

SYM

The SYM command in Report Writer forces a symmetric report, regardless of the data selection in the report script. Use SYM to change the symmetry of a report that Essbase would otherwise create as an asymmetric report.

Syntax

```
<SYM
```

Notes

The SYM 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

By default, 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 report script example is designed to work with Sample Basic, which is available in the gallery section of the Essbase file catalog. It generates an asymmetric report followed by a symmetric report.

```
<PAGE (Measures, Market)
Texas Sales
<ASYM
    <COLUMN (Scenario, Year)
        Actual Budget
        Jan Feb
<ROW (Product)
<IDESCENDANTS "100"
    !

<PAGE (Measures, Market)
Texas Sales
<SYM
    <COLUMN (Scenario, Year)
        Actual Budget
        Jan Feb
```

```
<ROW (Product)
<IDESCENDANTS "100"
!
```

The above report script example produces the following reports:

```

Sales Texas

Actual   Budget
  Jan     Feb
=====  =====
100-10      452      580
100-20      190      230
100-30      #Missing #Missing
  100      642      810

```

```

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

```

Related Topics

- [ASYM](#)
The Essbase Report Writer ASYM command causes a report to be printed in an asymmetric format.

TABDELIMIT

The TABDELIMIT Report Writer command in Essbase places tabs, rather than spaces, between columns. 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.

Syntax

```
{ TABDELIMIT }
```

Example

The following report scripts are designed for the Sample Basic cube, available in the gallery section of the Essbase file catalog.

```
<PAGE (Scenario)
<COLUMN (Year)
<ROW (Product, Market, Measures)
{Tabdelimit}
{ROWREPEAT}
```

```
<CHILDREN Year
<DIMBOTTOM Product
<DIMBOTTOM Market
<CHILD Profit
!
```

The above 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 example is the same report script without TABDELIMIT:

```
<PAGE (Scenario)
<COLUMN (Year)
<ROW (Product, Market, Measures)
{ROWREPEAT}
<CHILDREN Year
<DIMBOTTOM Product
<DIMBOTTOM Market
<CHILD Profit
!
```

Without TABDELIMIT, the report output displays as follows (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

The TEXT Report Writer command in Essbase 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 TEXT 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 a "*" 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. 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" ... ]}
```

Parameters

charPosition

Character position on the line to start the text specified in the next *text* argument. When multiple sets of *charPositions* and text can be specified, successive *charPositions* 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 *charPosition*.

text

Text to add to the report. Commas, tabs and multiple spaces are ignored. Maximum length: 500 characters.

Notes

- TEXT is an output command.
- *charPosition* must be an integer greater than or equal to zero, or the letter C for centered. (If you specify *charPosition* as zero, the line starts at the left margin.) You must specify a value for *charPosition*.
- 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 it.

Example

```
{ STARTHEADING
TEXT 1 "Prepared by:"
    14 "*USERNAME"
    C "My Report"
    65 "*PAGESTRING"
TEXT 65 "*DATE"
SKIP
ENDHEADING }
<PAGE (Measures)
Sales
<ASYM
<COLUMN (Scenario, Year)
Actual Budget
Jan Feb
<ROW (Product, Market)
<ACCOFF
"100-10"
"200-10"
"East"
"300-10"
"400-10"

{ SKIP 2 "Prepared by: " 18 "*USERNAME" }
{ TEXT 2 "Server Version: " 18 "*ARBOR" }
```



```
{ TEXT 2 "Application: " 18 "*APPNAME" }
{ TEXT 2 "Database: " 18 "*DBNAME" }
!
```

The above report script example produces the following report:

```
Prepared by: admin                My Report                Page: 1
                                      11/08/24

                                Sales

                                Actual   Budget
                                Jan       Feb
                                =====  =====

300-10                East                999       770
400-10                East                562       580

Server Version: Release 21.7.0 (ESB21.7.0.0.0)
Application:      Sample
Database:        Basic
```

Related Topics

- [SUPCOLHEADING](#)
The SUPCOLHEADING command in Essbase Report Writer suppresses display of default column headings.
- [SUPPAGEHEADING](#)
The Essbase Report Writer SUPPAGEHEADING command suppresses display of the page member heading whenever a heading is generated.

TODATE

The TODATE Report Writer command in Essbase 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)
```

Parameters

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")
```

Related Topics

- [ATTRIBUTE](#)
The Essbase Report Writer ATTRIBUTE command returns all base-dimension members associated with a specified attribute.
- [WITHATTR](#)
The WITHATTR Report Writer command specifies the characteristics of a base-dimension member that match the specified values in a report script. You must create attribute dimensions in the Essbase outline, and associate them with a base dimension, before you use WITHATTR.

TOP

The TOP Report Writer command in Essbase returns rows with the highest values of a specified data column.

Syntax

```
<TOP ([<rowgroupDimension>],] <rows>, <column>)
```

Parameters

<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>

@DATACOLUMN (<colNumber>) | @DATACOLUMN (<colNumber>)

where <colNumber> is the target column number; must be between 1 and the maximum number of columns in the report.

Notes

The TOP 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 configuration

Example

The following report script is designed for the Demo Basic cube, available in the gallery in the Essbase file catalog.

```
<Sym
<Column (Scenario, Year)
Actual Budget
Jan Dec
<Top ("Measures", 5, @DATACOLUMN(4))
<Row (Measures, Market, Product)
{SupMissingRows}

<Idescendants Profit
<Ichildren Market
<Idescendants Product
!
```

The report script above produces the following report:

			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

Related Topics

- **RESTRICT**
The Essbase Report Writer RESTRICT command specifies the conditions that the row must satisfy before it becomes part of a result set.
- **ORDERBY**
The Essbase Report Writer ORDERBY command orders the rows in a report according to data values in the specified columns.
- **BOTTOM**
The Essbase Report Writer BOTTOM command returns rows with the lowest values of a specified data column.

UCHARACTERS

The UCHARACTERS Report Writer command in Essbase 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"] }
```

Parameters**"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 report script example is based on Demo Basic, available in the gallery in the Essbase file catalog.

{UCHARACTERS} underlines all the characters in the previous (Television) row.

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual
```

```
    <COLUMN (Year)
    <CHILDREN Year
```

```
<ROW (Product)
Television
```

```
{ UCHARACTERS }
```

```
VCR
Compact_Disc
```

!

The report script example above 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

Related Topics

- [UCOLUMNS](#)
The UCOLUMNS Report Writer command in Essbase underlines all columns, including names and data, in the preceding row.
- [UDATA](#)
The UDATA Report Writer command in Essbase underlines data columns for a row, while not underlining the row name columns.
- [UNDERLINECHAR](#)
The UNDERLINECHAR Report Writer command in Essbase sets the default underline character displayed in a report.
- [UNDERSCORECHAR](#)
The UNDERSCORECHAR Report Writer command replaces the _ (underscore) character in an Essbase member name with another character.

UCOLUMNS

The UCOLUMNS Report Writer command in Essbase 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"] }
```

Parameters

"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 script example underlines all the columns in the previous row which is the Television row.

```
<PAGE (Market, Accounts, Scenario)
Chicago Sales Actual

      <COLUMN (Year)
      <ICHILDREN Year

<ROW (Product)
Television

{ UCOLUMNS }

VCR
Compact_Disc
      !
```

The above report script 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

Related Topics

- [UCHARACTERS](#)
The UCHARACTERS Report Writer command in Essbase underlines all non-blank characters in the preceding row.
- [UDATA](#)
The UDATA Report Writer command in Essbase underlines data columns for a row, while not underlining the row name columns.
- [UNDERLINECHAR](#)
The UNDERLINECHAR Report Writer command in Essbase sets the default underline character displayed in a report.

UDA

The UDA Report Writer command in Essbase selects and reports on members based on a common attribute, defined as a UDA (user-defined attribute).

Syntax

```
<UDA (dimName, udaStr)
```

Parameters

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.

Related Topics

- [SUDA](#)
The Essbase Report Writer SUDA command selects members based on a common attribute, defined as a UDA (user-defined attribute), along with their shared counterparts.

UDATA

The UDATA Report Writer command in Essbase 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"] }
```

Parameters

"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
!
```

The above report script 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
```


Related Topics

- [UCHARACTERS](#)
The UCHARACTERS Report Writer command in Essbase underlines all non-blank characters in the preceding row.
- [UNDERLINECHAR](#)
The UNDERLINECHAR Report Writer command in Essbase sets the default underline character displayed in a report.

UNAME

The UNAME Report Writer command in Essbase 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"] }
```

Parameters**"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 report script is designed for the Demo Basic cube, available in the gallery in the Essbase file catalog. The command { UNAME "*" } underlines with asterisks 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
!
```

The above report script 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

Related Topics

- [UCHARACTERS](#)

The UCHARACTERS Report Writer command in Essbase underlines all non-blank characters in the preceding row.

- [UDATA](#)

The UDATA Report Writer command in Essbase underlines data columns for a row, while not underlining the row name columns.

UNAMEONDIMENSION

The UNAMEONDIMENSION Report Writer command in Essbase underlines the row member names in a row whenever a member from the same dimension as the specified member changes.

Syntax

```
{ UNAMEONDIMENSION mbrName }
```

Parameters

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 report script example is based on Demo Basic.

```
<PAGE (Market, Accounts)
Chicago Sales
  <COLUMN (Scenario)
  Actual
```

```
<ROW (Year, Product)
{ UNAMEONDIMENSION Year }
<CHILDREN Year
<CHILDREN Audio
    !
```

The above report script 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
```

Related Topics

- [NOPAGEONDIMENSION](#)
The Essbase Report Writer NOPAGEONDIMENSION 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.
- [NOSKIPONDIMENSION](#)
The Essbase Report Writer NOSKIPONDIMENSION command prevents insertion of a new line when a member from the same dimension as the input member changes in a row of the report.
- [PAGEONDIMENSION](#)
The Essbase Report Writer PAGEONDIMENSION command 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.
- [SKIPONDIMENSION](#)
The Essbase Report Writer SKIPONDIMENSION command inserts a blank line when a member from the same dimension as the specified member changes on the next line in the report.

UNDERLINECHAR

The UNDERLINECHAR Report Writer command in Essbase 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 ["char"] }
```

Parameters

"char"

A single-byte character, enclosed in quotation marks, for the new underline character.

Notes

Double-byte characters are not supported.

Example

The following report script command sets the character used when underlining to a single dash.

```
{UNDERLINECHAR "-"} }
```

Related Topics

- [UCHARACTERS](#)
The UCHARACTERS Report Writer command in Essbase underlines all non-blank characters in the preceding row.
- [COLUMN](#)
The Essbase Report Writer COLUMN command defines the dimensions displayed as column members. Column members are displayed above data columns.
- [UDATA](#)
The UDATA Report Writer command in Essbase underlines data columns for a row, while not underlining the row name columns.

UNDERSCORECHAR

The UNDERSCORECHAR Report Writer command replaces the _ (underscore) character in an Essbase 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" }
```

Parameters**"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

The WIDTH Report Writer command in Essbase 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 ] ] ] }
```

Parameters**number**

New column width in characters.

column1 column2 columnN

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 *number*.

Notes

- The value of *number* 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 report script 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"
!
```

The report script example above resizes all data columns to a WIDTH of seven, and the row name label column (column 0) to a WIDTH of 20.

Scenario		Sales Illinois						
		Actual			Budget			
Feb	Mar	Jan	Feb	Mar	Jan	Feb	Mar	Jan
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								

Related Topics

- [NAMEWIDTH](#)
The Essbase Report Writer NAMEWIDTH command determines the width of all row name columns in the report.

WITHATTR

The WITHATTR Report Writer command specifies the characteristics of a base-dimension member that match the specified values in a report script. You must create attribute

dimensions in the Essbase outline, and associate them with a base dimension, before you use WITHATTR.

Syntax

```
<WITHATTR (dimName, "operator", value)
```

Parameters

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. 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:

Table 4-5 <WITHATTR Operator Examples

Operator	Example	Result
>	<WITHATTR(Population,">","18000000")	Returns New York, California, and Texas

Table 4-5 (Cont.) <WITHATTR Operator Examples

Operator	Example	Result
>=	<WITHATTR(Population,">=",10000000)	Returns New York, Florida, California, Texas, Illinois, and Ohio where 10,000,000 is not a numeric attribute member, but a constant
<	<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)
<ICHILDREN 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

Related Topics

- [<ATTRIBUTE](#)
The Essbase Report Writer ATTRIBUTE command returns all base-dimension members associated with a specified attribute.
- [<TODATE](#)
The TODATE Report Writer command in Essbase 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.

WITHATTREX

The WITHATTREX Report Writer command 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 Essbase outline, and associate them with a base dimension, before you use WITHATTREX in a report script.

Syntax

```
<WITHATTREX (dimName, "operator", value, options, startTuple[, endTuple])
```

Parameters

dimName

Single varying 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)

value

Value that, in combination with the operator, defines the condition that must be met. 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 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

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))
```

Related Topics

- [<ATTRIBUTEVA](#)
The Essbase Report Writer ATTRIBUTEVA command 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.
- [<PERSPECTIVE](#)
The Essbase Report Writer PERSPECTIVE command sets the perspective, a tuple or REALITY, for a varying attribute dimension for a report.
- [<TODATE](#)
The TODATE Report Writer command in Essbase 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.

ZEROTEXT

The ZEROTEXT Report Writer command in Essbase replaces zero data values with a text string, if a zero data value is output.

Syntax

```
{ ZEROTEXT [ "text" ] }
```

Parameters

text

Optional. String, in quotation marks, to use in place of 0.

Notes

All data values less than .0000000000001 and greater than -.0000000000001 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 -.

Related Topics

- [MISSINGTEXT](#)

The Essbase Report Writer MISSINGTEXT command replaces the default #MISSING label with your specified text, when a missing data value is generated on a line in the report.

Report Writer Limits

Learn about the Essbase report writer query limit, and limits for specific report writer commands.

Report Writer Query Limit

Report Writer supports regions that expand to no larger than $2^{64}-1$ cells.

If a report would expand to a larger-than-supported region, one of the following internal errors is returned:

```
1001632 "Internal Error: Mathematical operation resulted in integer overflow/underflow; reduce the size of the report region"
```

```
1200646 "Internal error: Set is too large to be processed. Set size exceeds 2^64 tuples"
```

```
1200713 "Internal Error: Mathematical operation results in integer overflow/underflow. Reduce the size of the query region"
```

```
1200762 "Internal Error: Mathematical operation results in wide integer overflow/underflow; reduce the size of the query region"
```

```
1204006 "Expanded query grid size does not fit into 64-bit integer. Change the query or simplify the outline model to include less formula dependencies."
```

Report Writer Command Limits

Command	Limit
CALCULATE COLUMN	No more than 50 column calculations can be defined at any one time in the report.
CALCULATE ROW	Limited to returning no more than 500 rows.