Essbase Analytic Services

Release 7.1

Database Administrator's Guides:

Volume I: Designing and Creating Analytic Services Databases

Volume II: Loading, Calculating, and Retrieving Data

Volume III: Optimization and System Administration

Volume IV: Creating, Calculating, and Managing Aggregate Storage Databases

Hyperion Solutions Corporation

Hyperion Solutions Corporation
1344 Crossman Avenue
Sunnyvale, California 94089

Printed in the U.S.A..

Essbase Analytic Services

Release 7.1

Database Administrator's Guide

Volume I: Designing and Creating Analytic Services Databases

Hyperion Solutions Corporation

Hyperion Solutions Corporation
1344 Crossman Avenue
Sunnyvale, California 94089

Printed in the U.S.A..

Contents

Preface

Welcome to the *Essbase Analytic Services Database Administrator's Guide*. This preface discusses the following topics:

Purpose

This guide provides you with all the information that you need to implement, design, and maintain an optimized Essbase Analytic Services multidimensional database. It explains the Analytic Services features and options, and contains the concepts, processes, and examples that you need to use the software.

Audience

This guide is for database administrators or system administrators who are responsible for designing, creating, and maintaining applications, databases, and database objects (for example, data load rules, and calculation scripts).

Document Structure

This document contains the following information:

- Chapters 1 through 14 (Volume I) contain information on designing and creating Analytic Services databases, including information on converting currencies, partitioning applications, and accessing relational data using the hybrid analysis feature.
- Chapters 15 through 34 (Volume II) contain information on building dimensions and loading data, calculating data, and retrieving data.
- Chapters 35 through 56 (Volume III) contain information on designing and managing a security system, maintaining, backing up, and optimizing Analytic Services databases, and includes a glossary of key terms and their definitions.
- Chapters 57 through 60 (Volume IV) contain information on aggregate storage databases.

Where to Find Documentation

All Analytic Services documentation is accessible from the following locations:

- The HTML Information Map is located at *`installation_directory`*`\docs\esb_infomap.htm`, where *installation_directory* is the directory in which you have installed the Analytic Services documentation.
- The Hyperion Solutions Web site is located at `http://www.hyperion.com`.
- The Hyperion Download Center can be accessed from `http://hyperion.subscribenet.com` or from `http://www.hyperion.com`.

➤ To access documentation from the Hyperion Solutions Web site:

1. Log on to `http://www.hyperion.com`.
2. Select the **Support** link and type your username and password to log on.

   Note: New users must register to receive a username and password.

3. Click the **Hyperion Download Center** link and follow the on-screen instructions.

➤ To access documentation from the Hyperion Download Center:

1. Log on to `http://hyperion.subscribenet.com`.
2. In the **Login ID** and **Password** text boxes, enter your assigned login ID name and password. Then click **Login**.
3. If you are a member on multiple Hyperion Download Center accounts, select the account that you want to use for the current session.
4. Follow the on-screen instructions.
5. Perform one of the following actions:
   - To access documentation online, from the **Product List**, select the appropriate product and follow the on-screen instructions.
   - To order printed documentation, from the **Information** section in the left frame, select **Order Printed Documentation**, then follow the on-screen instructions

➤ To order printed documentation if you do not have access to the Hyperion Download Center:

- In the United States, call Hyperion Solutions Customer Support at 877-901-4975.
- From outside the United States, including Canada, call Hyperion Solutions Customer Support at 203-703-3600. Clients who are not serviced by support from North America should call their local support centers.

Conventions

The following table shows the conventions that are used in this document:

Table i: Conventions Used in This Document

Item	Meaning
➤	Arrows indicate the beginning of procedures consisting of sequential steps or one-step procedures.
Brackets []	In examples, brackets indicate that the enclosed elements are optional.

Table i: Conventions Used in This Document (Continued)

Item	Meaning
Bold	Bold in procedural steps highlights major interface elements.
CAPITAL LETTERS	Capital letters denote commands and various IDs. (Example: CLEARBLOCK command)
Ctrl + 0	Keystroke combinations shown with the plus sign (+) indicate that you should press the first key and hold it while you press the next key. Do not type the plus sign.
`Example text`	Courier font indicates that the example text is code or syntax.
`Courier italics`	Courier italic text indicates a variable field in command syntax. Substitute a value in place of the variable shown in Courier italics.
ARBORPATH	When you see the environment variable *ARBORPATH* in italics, substitute the value of ARBORPATH from your site.
n, *x*	Italic *n* stands for a variable number; italic *x* can stand for a variable number or an alphabet. These variables are sometimes found in formulas.
Ellipses (...)	Ellipsis points indicate that text has been omitted from an example.
Mouse orientation	This document provides examples and procedures using a right-handed mouse. If you use a left-handed mouse, adjust the procedures accordingly.
Menu options	Options in menus are shown in the following format. Substitute the appropriate option names in the placeholders, as indicated. ***Menu name*** > ***Menu command*** > ***Extended menu command*** For example: **1.** Select **File** > **Desktop** > **Accounts**.

Additional Support

In addition to providing documentation and online help, Hyperion offers the following product information and support. For details on education, consulting, or support options, click the Services link on the Hyperion Web site at http://www.hyperion.com.

Education Services

Hyperion offers instructor-led training, custom training, and eTraining covering all Hyperion applications and technologies. Training is geared to administrators, end users, and information systems (IS) professionals.

Consulting Services

Experienced Hyperion consultants and partners implement software solutions tailored to clients' particular reporting, analysis, modeling, and planning requirements. Hyperion also offers specialized consulting packages, technical assessments, and integration solutions.

Technical Support

Hyperion provides enhanced electronic-based and telephone support to clients to resolve product issues quickly and accurately. This support is available for all Hyperion products at no additional cost to clients with current maintenance agreements.

Documentation Feedback

Hyperion strives to provide complete and accurate documentation. We value your opinions on this documentation and want to hear from you. Send us your comments by clicking the link for the Documentation Survey, which is located on the Information Map for your product.

Part I

Understanding Essbase Analytic Services

Part I introduces you to Essbase Analytic Services by describing general online analytical processing (OLAP) concepts, basic multidimensional concepts, the Analytic Services architecture, and how to design both single server and partitioned applications. Part I contains the following chapters:

- Chapter 1, "Introducing Hyperion Essbase," describes the parts of Analytic Services, high-level Analytic Services functionality, key architectural features, and how Analytic Services works in a client-server environment.
- Chapter 2, "Understanding Multidimensional Databases," introduces you to basic multidimensional concepts and terminology, including dimensions and members, data values, and hierarchies.
- Chapter 4, "Basic Architectural Elements," describes how the Analytic Services architecture stores and retrieves information.
- Chapter 3, "Quick Start for Implementing Analytic Services," provides a high-level process map for implementing Analytic Services in your organization with cross references to further information.
- Chapter 5, "Case Study: Designing a Single-Server, Multidimensional Database," uses the Sample Basic database to present rules you should use to design a single-server, multidimensional database solution.
- Chapter 6, "About Essbase Administration Services," describes the client interfaces to Analytic Services, focussing on Administration Services, the new cross-platform administration tool for Analytic Services.

Chapter 1

Introducing Hyperion Essbase

This chapter provides an architectural overview of the product components and introduces the key features of Essbase products, including:

- Essbase Analytic Services
- Essbase Deployment Services
- Essbase Administration Services
- Essbase Spreadsheet Services
- Essbase Integration Services

Essbase products provide companies with the ability to deliver critical business information to the right people when they need it. With Essbase, companies quickly leverage and integrate data from multiple existing data sources and distribute filtered information to end-user communities in the format that best meets the users' needs. Users interact and intuitively explore data in real-time and along familiar business dimensions, enabling them to perform speed-of-thought analytics.

This chapter contains the following sections:

Key Features

Essbase products provide the analytic solution that integrates data from multiple sources and meets the needs of users across an enterprise. Essbase products enable the quick and easy implementation of solutions, add value to previously inaccessible data, and transform data into actionable information.

Integration with Existing Infrastructure

Essbase products integrate with your existing business intelligence infrastructure. Essbase products meet the enterprise analytic demands of users for critical business information with a minimum of information technology (IT) overhead and thus enable organizations to realize maximum return on their existing IT investments:

- Provides an extensible architecture
- Supports a comprehensive range of data sources, hardware and operating system platforms, access interfaces, and development languages
- Enables analytic applications to be deployed across a local or wide area network and across an intranet or Internet

Data Integration

Essbase products enable organizations to leverage data in their data warehouses, legacy systems, online transaction processing (OLTP) systems, enterprise resource planning (ERP) systems, e-business systems, customer relationship management (CRM) applications, Web log files and other external data sources. For database integration, Essbase Integration Services provides a suite of graphical tools, data integration services, and a metadata catalog that tie into relational databases or data warehouse environments (including SAP BW).

Ease of Server and Database Administration

Essbase products provide a cross-platform administration console. The console gives you detailed control over the Essbase environment:

- You can manage multiple servers and databases.
- You can use MaxL, a syntactical language command shell with a PERL extension module, to automate batch maintenance.

Mission Critical Applications in Web-based Environments

A middle-tier framework extends the power of Essbase products by creating a Web-enabled, distributed platform for Essbase applications hence serving the analysis needs of large numbers of users in Web-based environments. Essbase Deployment Services provide connection pooling, clustering, and failover support, which extend the scalability and reliability of the platform and support mission-critical applications in a 24 x 7 environment.

Powerful Querying

Large communities of business users can interact with data in real time, analyzing business performance at the speed of thought. Using Essbase products you can organize and present data along familiar business dimensions, thus enabling users to view and explore the data intuitively and to turn the data into actionable information.

Calculations

Essbase Analytic Services includes powerful calculation features for demanding analytic requirements. A rich library of functions makes it easy to define advanced and sophisticated business logic and relationships. Analytic Services gives users the flexibility to build, customize and extend the calculator through custom-defined macros and functions, as well as the ability to span calculations across databases. On multiprocessor systems, a database administrator can configure a single calculation request to use multiple threads to accomplish the calculation, providing enhanced calculation speed.

Aggregate storage databases provide an alternative to block storage databases, and enable dramatic improvements in database aggregation time for certain types of applications.

Write-Back and Security

Analytic Services provides unique multi-user read and write capabilities, including data update and multi-user recalculation. Business users with front-end tools can write data back to a server and recalculate the data on a server using calculation scripts – key functionality to support sophisticated modeling and planning applications.

The robust, multi-level security model provides server-, database-, and cell-level security. Full control of data access, views, and write capabilities are managed through administration. Integration with external authentication systems, such as Lightweight Directory Access Protocol (LDAP), allows the quick creation of security models and reduces the time and the cost of application development and maintenance.

Ease of Development

Analytic Services offers many key advantages to help users develop effective multi-dimensional applications. Users can:

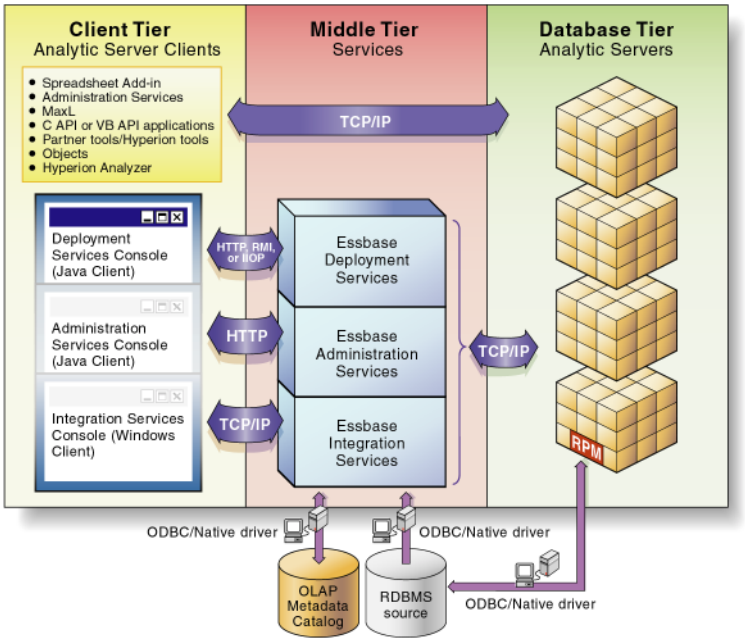
- Design and manage applications using a graphical interface to control most server functions.
- Quickly add dimensions, change calculations, and modify hierarchies to reflect new business developments. In addition, the dynamic dimension builder automatically defines and dynamically loads large amounts of data, including data from spreadsheets, flat files, and supported relational database tables directly into a database.
- Define key calculations without having to write a program.
- Define security for individuals and groups and customize views and retrieval procedures for each user without writing a program.

For information about supported applications, operating systems, and networking protocols, see the *Essbase Analytic Services Installation Guide*.

Essbase Product Components

Essbase products incorporate powerful architectural features to handle a wide range of analytic applications across large multi-user environments. Figure 1 provides a high-level view of the information flow between the source data and the product components.

Figure 1: High-level Information Flow Between Product Components



Note: For information on Hyperion Objects, Hyperion Analyzer, and Hyperion Application Link (HAL) visit our website at `www.hyperion.com`.

Analytic Services

Analytic Services—a multi-threaded OLAP database software that takes advantage of symmetric multi processing hardware platforms—is based upon Web-deployable, thin-client architecture. The server acts as a shared resource, handling all data storage, caching, calculations, and data security. The Analytic Server client needs only to retrieve and view data that resides on a server.

All Analytic Services application components, including database outlines and calculation scripts, application control, and multi-dimensional database information, reside on a server. With Analytic Services you can configure server disk storage to span multiple disk drives, enabling you to store large databases. Analytic Services requires a server to run a multi-threaded operating system so a server can efficiently manage multiple, simultaneous requests. A server also runs a server agent process that acts as a traffic coordinator for all user requests to applications.Aggregate storage databases—using a new storage kernel for multidimensional databases—provide an alternative to block storage databases, and enable dramatic increases in database dimensionality. Using aggregate storage, Analytic Services serves a wide range of analytic needs—financial analysis, planning, budgeting, sales analysis, marketing analysis, supply chain analysis, profitability analytics—all from a single analytic infrastructure.

MaxL—a multidimensional database access language that is part of Analytic Server—provides a flexible way to automate Analytic Services administration and maintenance tasks.

See `dev.hyperion.com` or the *Essbase Analytic Services Installation Guide* for information on the supported operating systems, and for specific information about server configuration requirements.

Administration Services

Administration Services—the database and system administrators' interface to Analytic Services—provides a single-point-of-access console to multiple Analytic Servers. Using Administration Services you can design, develop, maintain, and manage multiple Analytic Servers, applications, and databases. You can preview data from within the console, without having to open a client application such as Spreadsheet Add-in. You can also use custom Java plug-ins to leverage and extend key functionality.

Deployment Services

Deployment Services allows multiple instances of Analytic Server to run on multiple machines, while serving the user as one logical unit and removing and single point of failure. Deployment Services enables database clustering with load balancing and fail-over capabilities.

Spreadsheet Products and Hyperion Analyzer

Hyperion Analyzer, Spreadsheet Services, and Spreadsheet Add-in—the business users' interface to Analytic Services—provide interactive analysis and delivery of corporate information to diverse user communities. Hyperion Analyzer, Spreadsheet Services, and Spreadsheet Add-in provide out-of-the-box solutions and easy-to-use personalization and visualization tools allow you to create intuitive, Web-based analysis and reporting from ERP systems, relational, multidimensional and other data sources.

Integration Services

Integration Services—an optional product component—provides a metadata-driven environment to bridge the gap between data stored in Analytic Services databases and detailed data stored in relational databases. The Hybrid Analysis feature gives business users more detail for decision-making and IT managers more modularity in designing and maintaining large-scale analytic applications. Hybrid Analysis allows portions of Analytic Services databases to be stored in a relational database. This relational stored data is mapped to the appropriate Analytic Services hierarchies. HAL (Hyperion Application Link) is an application integration and business process automation tool that allows bi-directional information exchange between transaction processing applications, desktop applications, and Hyperion Business Performance Management applications.

Application Programming Interface (API)

Analytic Services API—the developers' interface to Analytic Services—allows you to create customized applications. The *API Reference* provides a complete listing of API functions, platforms, and supported compilers.

Developer Products

Essbase developer products enable the rapid creation, management and deployment of tailored enterprise analytic applications—with or without programming knowledge.

The products (for example, Application Builder, and Hyperion Objects) provide a comprehensive set of application programming interfaces, drag and drop components and services.

Data Mining

Data Mining—an optional product component of Analytic Services—shows you hidden relationships and patterns in your data, enabling you to make better business decisions. Using Data Mining you can plug in various data mining algorithms, build models, and then apply them to existing Analytic Services applications and databases.

Chapter 2

Understanding Multidimensional Databases

Essbase Analytic Services contains multidimensional databases that support analysis and management reporting applications. This chapter discusses multidimensional concepts and terminology.

Note: The information in this chapter is designed for block storage databases. Some of the information is not relevant to aggregate storage databases. For detailed information on the differences between aggregate and block storage, see Chapter 57, "Comparison of Aggregate and Block Storage."

This chapter contains the following topics:

OLAP and Multidimensional Databases

Online analytical processing (OLAP) is a multidimensional, multi-user, client-server computing environment for users who need to analyze enterprise data. OLAP applications span a variety of organizational functions. Finance departments use OLAP for applications such as budgeting, activity-based costing (allocations), financial performance analysis, and financial modeling. Sales departments use OLAP for sales analysis and forecasting. Among other applications, marketing departments use OLAP for market research analysis, sales forecasting, promotions analysis, customer analysis, and market/customer segmentation. Typical manufacturing OLAP applications include production planning and defect analysis.

Important to all of the applications mentioned in the previous paragraph is the ability to provide managers with the information they need to make effective decisions about an organization's strategic directions. A successful OLAP application provides information as needed, that is, it provides “just-in-time” information for effective decision-making.

Providing “just-in-time” information requires more than a base level of detailed data. “Just-in-time” information is computed data that usually reflects complex relationships and is often calculated on the fly. Analyzing and modeling complex relationships are practical only if response times are consistently short. In addition, because the nature of data relationships may not be known in advance, the data model must be flexible. A truly flexible data model ensures that OLAP systems can respond to changing business requirements as needed for effective decision making.

Although OLAP applications are found in widely divergent functional areas, they all require the following key features:

- Multidimensional views of data
- Calculation-intensive capabilities
- Time intelligence

Key to OLAP systems are multidimensional databases. Multidimensional databases not only consolidate and calculate data; they also provide retrieval and calculation of a variety of data subsets. A multidimensional database supports multiple views of data sets for users who need to analyze the relationships between data categories. For example, a marketing analyst might want answers to the following questions:

- How did Product A sell last month? How does this figure compare to sales in the same month over the last five years? How did the product sell by branch, region, and territory?
- Did this product sell better in particular regions? Are there regional trends?
- Did customers return Product A last year? Were the returns due to product defects? Did the company manufacture the products in a specific plant?
- Did commissions and pricing affect how salespeople sold the product? Did particular salespeople do a better job of selling the product?

With a multidimensional database, the number of data views is limited only by the database outline, the structure that defines all elements of the database. Users can pivot the data to see information from a different viewpoint, drill down to find more detailed information, or drill up to see an overview.

Dimensions and Members

This section introduces the concepts of outlines, dimensions and members within a multidimensional database. If you understand dimensions and members, you are well on your way to understanding the power of a multidimensional database.

A dimension represents the highest consolidation level in the database outline. The database outline presents dimensions and members in a tree structure to indicate a consolidation relationship. For example, in Figure 2, Time is a dimension and Qtr1 is a member.

Analytic Services has two types of dimensions: standard dimensions and attribute dimensions.

Standard dimensions represent the core components of a business plan and often relate to departmental functions. Typical standard dimensions are Time, Accounts, Product Line, Market, and Division. Dimensions change less frequently than members.

Attribute dimensions are a special type of dimension that are associated with standard dimensions. Through attribute dimensions, you group and analyze members of standard dimensions based on the member attributes (characteristics). For example, you can compare the profitability of non-caffeinated products that are packaged in glass to the profitability of non-caffeinated products that are packaged in cans.

Members are the individual components of a dimension. For example, Product A, Product B, and Product C might be members of the Product dimension. Each member has a unique name. A dimension can contain an unlimited number of members. Analytic Services can store the data associated with a member (referred to as a stored member in this chapter) or it can dynamically calculate the data when a user retrieves it.

Outline Hierarchies

All Analytic Services database development begins with creating a database outline. A database *outline* accomplishes the following:

- Defines the structural relationships between members in an Analytic Services database
- Organizes all the data in the database
- Defines the consolidations and mathematical relationships between items

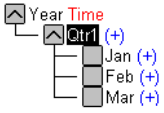
Analytic Services uses the concept of members to represent data hierarchies. Each dimension consists of one or more members. The members, in turn, may consist of other members. When you create a dimension, you tell Analytic Services how to *consolidate* the values of its individual members. Within the tree structure of the database outline, a consolidation is a group of members in a branch of the tree.

For example, many businesses summarize their data monthly, roll up the monthly data to obtain quarterly figures, and roll up the quarterly data to obtain annual figures. Businesses may also summarize data by zip code, by city, state, and country. Any dimension can be used to consolidate data for reporting purposes.

In the Sample Basic database included with Analytic Server, for example, the Year dimension consists of five members: Qtr1, Qtr2, Qtr3, and Qtr4, each storing data for an individual quarter, plus Year, storing summary data for the entire year. Qtr1 consists of four members: Jan, Feb, and Mar, each storing data for an individual month, plus Qtr1, storing summary data for the entire quarter. Likewise, Qtr2, Qtr3, and Qtr4 consist of the members that represent the individual months plus the member that stores the quarterly totals.

The database outline in Figure 2 uses a hierarchical structure to represent the data consolidations and relationships in Qtr1.

Figure 2: Hierarchical Structure



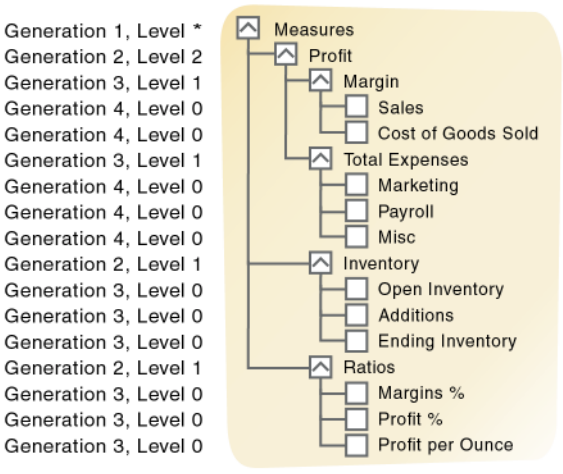
Some dimensions consist of relatively few members, while others may have hundreds or even thousands of members. Analytic Services does not limit the number of members within a dimension and enables the addition of new members as needed.

Dimension and Member Relationships

Analytic Services uses the terms defined in the following sections to describe a database outline. These terms are used throughout Analytic Services documentation.

Analytic Services uses hierarchical and family history terms to describe the roles and relationships of the members in an outline. You can describe the position of the members of the branches in Figure 3 in several ways.

Figure 3: Member Generation and Level Numbers



* The level of Measures depends on the branch

Parents, Children, and Siblings

Figure 3 illustrates the following parent, child, and sibling relationships:

- A *parent* is a member that has a branch below it. For example, Margin is a parent member for Sales and Cost of Goods Sold.
- A *child* is a member that has a parent above it. For example, Sales and Cost of Goods Sold are children of the parent Margin.
- *Siblings* are child members of the same immediate parent, at the same generation. For example, Sales and Cost of Goods Sold are siblings (they both have the parent Margin), but Marketing (at the same branch level) is not a sibling because its parent is Total Expenses.

Descendants and Ancestors

Figure 3 illustrates the following descendant and ancestral relationships:

- *Descendants* are all members in branches below a parent. For example, Profit, Inventory, and Ratios are descendants of Measures. The children of Profit, Inventory, and Ratios are also descendants of Measures.
- *Ancestors* are all members in branches above a member. For example, Margin, Profit, and Measures are ancestors of Sales.

Roots and Leaves

Figure 3 illustrates the following root and leaf member relationships:

- The *root* is the top member in a branch. Measures is the root for Profit, Inventory, Ratios, and the children of Profit, Inventory, and Ratios.
- *Leaf* members have no children. They are also referred to as detail members, level 0 members, and leaf nodes. For example, Opening Inventory, Additions, and Ending Inventory are leaf members.

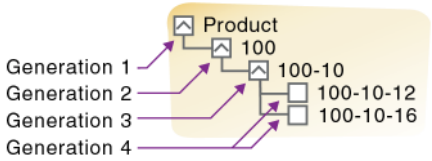
Generations and Levels

Figure 3 illustrates the following generations levels:

- *Generation* refers to a consolidation level within a dimension. A root branch of the tree is generation 1. Generation numbers increase as you count from the root toward the leaf member. In Figure 3, Measures is generation 1, Profit is generation 2, and Margin is generation 3. All siblings of each level belong to the same generation; for example, both Inventory and Ratios are generation 2.

  Figure 4 shows part of the Product dimension with its generations numbered.

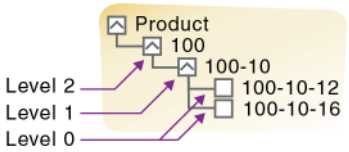
  Figure 4: Generations



- *Level* also refers to a branch within a dimension; however, levels reverse the numerical ordering that Analytic Services uses for generations. The levels count up from the leaf member toward the root. The root level number varies depending on the depth of the branch. In the example in Figure 3, Sales and Cost of Goods Sold are level 0. All other leaf members are also level 0. Margin is level 1, and Profit is level 2. Notice that the level number of Measures varies depending on the branch. For the Ratios branch, Measures is level 2. For the Total Expenses branch, Measures is level 3.

  Figure 5 shows part of the Product dimension with its levels numbered.

  Figure 5: Levels



Generation and Level Names

To make your reports easier to maintain, you can assign a name to a generation or level and then use the name as a shorthand for all members in that generation or level. Because changes to an outline are automatically reflected in a report, when you use generation and level names, you do not need to change the report if a member name is changed or deleted from the database outline.

Standard Dimensions and Attribute Dimensions

Analytic Services has two types of dimensions: standard dimensions and attribute dimensions. This chapter primarily considers standard dimensions because Analytic Services does not allocate storage for attribute dimension members. Instead it dynamically calculates the members when the user requests data associated with them.

An attribute dimension is a special type of dimension that is associated with a standard dimension. For comprehensive discussion of attribute dimensions, see Chapter 10, “Working with Attributes.”

Sparse and Dense Dimensions

Most data sets of multidimensional applications have two characteristics:

- Data is not smoothly and uniformly distributed.
- Data does not exist for the majority of member combinations. For example, all products may not be sold in all areas of the country.

Analytic Services maximizes performance by dividing the standard dimensions of an application into two types: *dense dimensions* and *sparse dimensions*. This division allows Analytic Services to cope with data that is not smoothly distributed. Analytic Services speeds up data retrieval while minimizing the memory and disk requirements.

Most multidimensional databases are inherently sparse: they lack data values for the majority of member combinations. A sparse dimension is a dimension with a low percentage of available data positions filled.

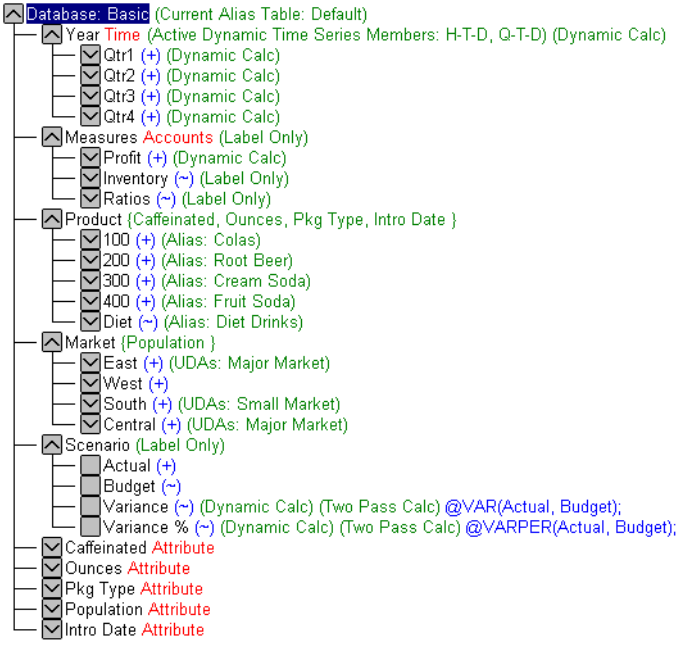
For example, the Sample Basic database shown in Figure 6 includes the Year, Product, Market, Measures, and Scenario dimensions. Product represents the product units, Market represents the geographical regions in which the products are sold, and Measures represents the accounts data. Because not every product is sold in every market, Market and Product are chosen as sparse dimensions.

Most multidimensional databases also contain dense dimensions. A dense dimension is a dimension with a high probability that one or more data points is occupied in every combination of dimensions. For example, in the Sample Basic database, accounts data exists for almost all products in all markets, so Measures is chosen as a dense dimension. Year and Scenario are also chosen as dense dimensions. Year represents time in months, and Scenario represents whether the accounts values are budget or actual values.

In “Sample Basic Database Outline” on page 39, Caffeinated, Intro Date, Ounces, and Pkg Type are attribute dimensions that are associated with the Product dimension. Population is an attribute dimension that is associated with the Market dimension. Members of attribute dimensions describe characteristics of the members of the dimensions with which they are associated. For example, each product has a size in ounces. Attribute dimensions are always sparse dimensions and must be associated with a sparse standard dimension. Analytic Services does

not store the data for attribute dimensions, Analytic Services dynamically calculates the data when a user retrieves it. For a comprehensive discussion about attribute dimensions, see Chapter 10, "Working with Attributes."

Figure 6: Sample Basic Database Outline



Selection of Dense and Sparse Dimensions

In most data sets, existing data tends to follow predictable patterns of density and sparsity. If you match patterns correctly, you can store the existing data in a reasonable number of fairly dense data blocks, rather than in many highly sparse data blocks.

Analytic Services can make recommendations for the sparse-dense configuration of dimensions based on the following factors:

- The time and accounts tags on dimensions
- The probable size of the data blocks
- Characteristics that you attribute to the dimensions in this dialog box

You can apply a recommended configuration or you can turn off automatic configuration and manually set the sparse or dense property for each dimension. Attribute dimensions are always sparse dimensions. Keep in mind that you can associate attribute dimensions only with sparse standard dimensions.

Note: The automatic configuration of dense and sparse dimensions provides only an estimate. It cannot take into account the nature of the data you will load into your database or multiple user considerations.

Dense-Sparse Configuration for Sample Basic

Consider the Sample Basic database that is provided with Analytic Services. The Sample Basic database represents data for The Beverage Company (TBC).

TBC does not sell every product in every market; therefore, the data set is reasonably sparse. Data values do not exist for many combinations of members in the Product and Market dimensions. For example, if Caffeine Free Cola is not sold in Florida, then data values do not exist for the combination Caffeine Free Cola (100-30)->Florida.

However, consider combinations of members in the Year, Measures, and Scenario dimensions. Data values almost always exist for some member combinations on these dimensions. For example, data values exist for the member combination Sales->January->Actual because at least some products are sold in January.

The sparse-dense configuration of the standard dimensions in the Sample Basic database may be summarized as follows:

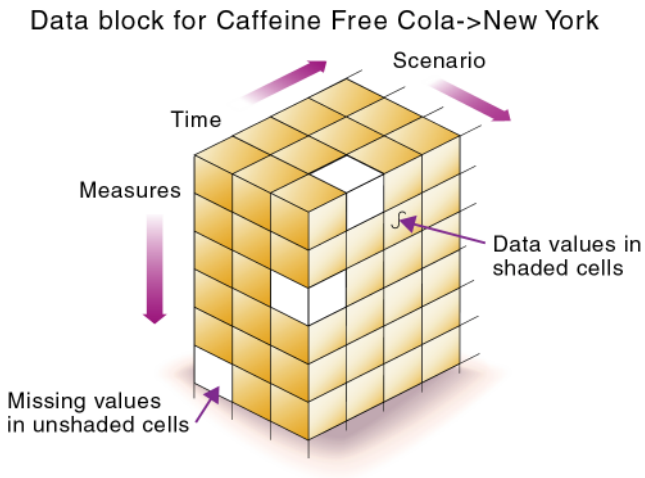
- The sparse standard dimension are Product and Market.
- The dense standard dimensions are Year, Measures, and Scenario.

Analytic Services creates a *data block* for each unique combination of members in the Product and Market dimensions (for more information on data blocks, see "Data Storage" on page 44). Each data block represents data from the dense dimensions. The data blocks are likely to have few empty cells.

For example, consider the sparse member combination Caffeine Free Cola (100-30), New York, illustrated by Figure 7:

- If accounts data (represented by the Measures dimension) exists for this combination for January, it probably exists for February and for all members in the Year dimension.
- If a data value exists for one member on the Measures dimension, then it is likely that other accounts data values exist for other members in the Measures dimension.
- If Actual accounts data values exist, then it is likely that Budget accounts data values exist.

Figure 7: Dense Data Block for Sample Basic Database

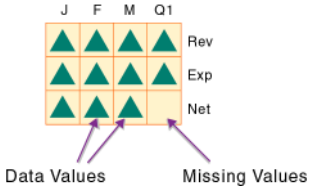


Dense and Sparse Selection Scenario

Consider a database with four standard dimensions: Time, Accounts, Region, and Product. In the following example, Time and Accounts are dense dimensions, and Region and Product are sparse dimensions.

The two-dimensional data blocks shown in Figure 8 represent data values from the dense dimensions: Time and Accounts. The members in the Time dimension are J, F, M, and Q1. The members in the Accounts dimension are Rev, Exp, and Net.

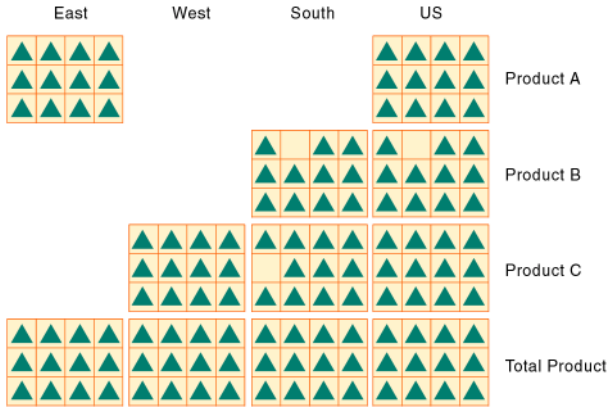
Figure 8: Two-dimensional Data Block for Time and Accounts



Analytic Services creates data blocks for combinations of members in the sparse standard dimensions (providing at least one data value exists for the member combination). The sparse dimensions are Region and Product. The members of the Region dimension are East, West, South, and Total US. The members in the Product dimension are Product A, Product B, Product C, and Total Product.

Figure 9 shows 11 data blocks. No data values exist for Product A in the West and South, for Product B in the East and West, and for Product C in the East. Therefore, Analytic Services has not created data blocks for these member combinations. The data blocks that Analytic Services has created have very few empty cells.

Figure 9: Data Blocks Created for Sparse Members on Region and Product

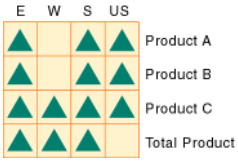


This example effectively concentrates all the sparseness into the index and concentrates all the data into fully utilized blocks. This configuration provides efficient data storage and retrieval.

Now consider a reversal of the dense and sparse dimension selections. In the following example, Region and Product are dense dimensions, and Time and Accounts are sparse dimensions.

As shown in Figure 10, the two-dimensional data blocks represent data values from the dense dimensions: Region and Product.

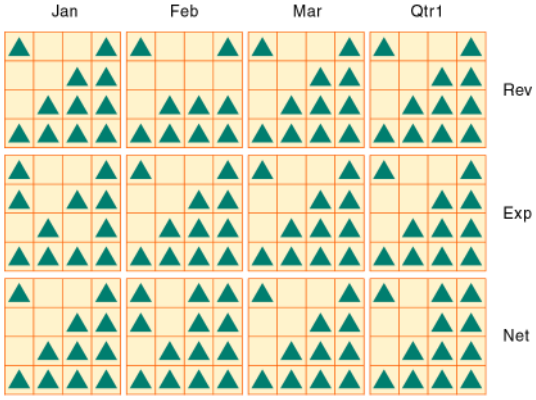
Figure 10: Two-Dimensional Data Block for Region and Product



Analytic Services creates data blocks for combinations of members in the sparse standard dimensions (providing at least one data value exists for the member combination). The sparse standard dimensions are Time and Accounts.

Figure 11 shows 12 data blocks. Data values exist for all combinations of members in the Time and Accounts dimensions; therefore, Analytic Services creates data blocks for all the member combinations. Because data values do not exist for all products in all regions, the data blocks have many empty cells. Data blocks with many empty cells store data inefficiently.

Figure 11: Data Blocks Created for Sparse Members on Time and Accounts



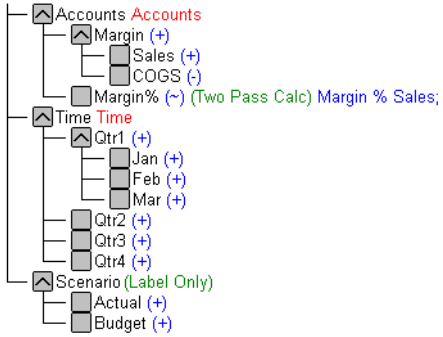
Data Storage

This topic describes how data is stored in a multidimensional database. Each *data value* is stored in a single cell in the database. You refer to a particular data value by specifying its coordinates along *each* standard dimension.

Note: Analytic Services does not store data for attribute dimensions. Analytic Services dynamically calculates attribute dimension data when a user retrieves the data.

Consider the simplified database shown in Figure 12.

Figure 12: A Multidimensional Database Outline



This database has three dimensions: Accounts, Time, and Scenario:

- The Accounts dimension has four members: Sales, COGS, Margin, and Margin%.
- The Time dimension has four quarter members, and Qtr1 has three month members

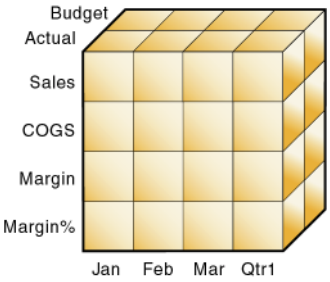
  Note: Figure 13 shows only Qtr1 and its members.

- The Scenario dimension has two child members: Budget for budget values and Actual for actual values.

Data Values

The intersection of one member from one dimension with one member from each of the other dimensions represents a data value. The example in Figure 13 has three dimensions; thus, the dimensions and data values in the database can be represented in a cube.

Figure 13: Three-Dimensional Database



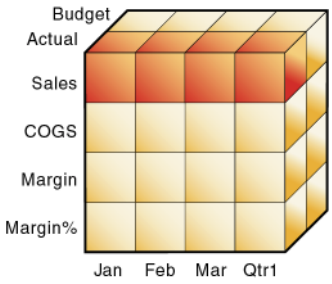
The shaded cells in Figure 14 illustrate that when you specify Sales, you are specifying the portion of the database containing eight Sales values.

Figure 14: Sales Slice of the Database

Slicing a database amounts to fixing one or more dimensions at a constant value while allowing the other dimensions to vary.

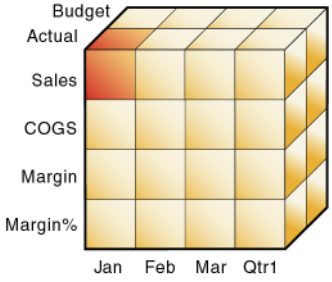
When you specify Actual Sales, you are specifying the four Sales values where Actual and Sales intersect as shown by the shaded area in Figure 15.

Figure 15: Actual, Sales Slice of the Database



A data value is stored in a single cell in the database. To refer to a specific data value in a multidimensional database, you specify its member on each dimension. In Figure 16, the cell containing the data value for Sales, Jan, Actual is shaded. The data value can also be expressed using the cross-dimensional operator (->) as Sales -> Actual -> Jan.

Figure 16: Sales -> Jan -> Actual Slice of the Database



Data Blocks and the Index System

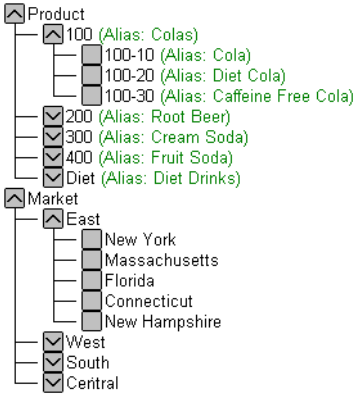
Analytic Services uses two types of internal structures to store and access data: data blocks and the index system.

Analytic Services creates a *data block* for each unique combination of sparse standard dimension members (providing that at least one data value exists for the sparse dimension member combination). The data block represents all the dense dimension members for its combination of sparse dimension members.

Analytic Services creates an *index entry* for each data block. The index represents the combinations of sparse standard dimension members. It contains an entry for each unique combination of sparse standard dimension members for which at least one data value exists.

For example, in the Sample Basic database outline shown in Figure 17, Product and Market are sparse dimensions.

Figure 17: Product and Market Dimensions from the Sample Basic Database



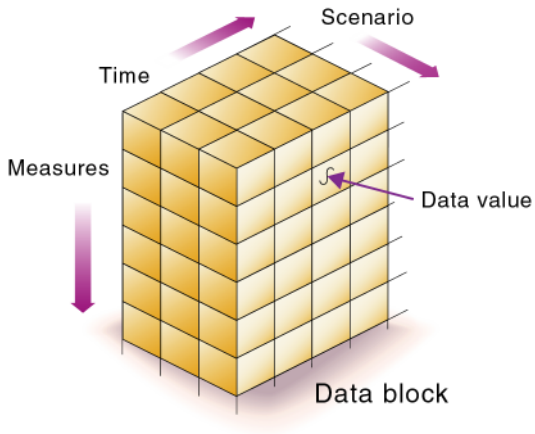
If data exists for Caffeine Free Cola in New York, then Analytic Services creates a data block and an index entry for the sparse member combination of Caffeine Free Cola (100-30) -> New York. If Caffeine Free Cola is *not* sold in Florida, then Analytic Services does *not* create a data block or an index entry for the sparse member combination of Caffeine Free Cola (100-30) -> Florida.

The data block Caffeine Free Cola (100-30) -> New York represents all the Year, Measures, and Scenario dimensions for Caffeine Free Cola (100-30) -> New York.

Each unique data value can be considered to exist in a cell in a data block. When Analytic Services searches for a data value, it uses the index to locate the appropriate data block.Then, within the data block, it locates the cell containing the data value. The index entry provides a pointer to the data block. The index handles sparse data efficiently because it includes only pointers to existing data blocks.

Figure 18 shows part of a data block for the Sample Basic database. Each dimension of the block represents a dense dimension in the Sample Basic database: Time, Measures, and Scenario. A data block exists for each unique combination of members of the Product and Market sparse dimensions (providing that at least one data value exists for the combination).

Figure 18: Part of a Data Block for the Sample Basic Database



Each data block is a multidimensional array that contains a fixed, ordered location for each possible combination of dense dimension members. Accessing a cell in the block does not involve sequential or index searches. The search is almost instantaneous, resulting in optimal retrieval and calculation speed.

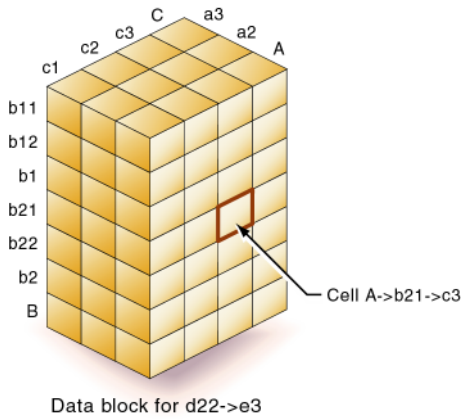
Analytic Services orders the cells in a data block according to the order of the members in the dense dimensions of the database outline.

```
A (Dense)
   a1
   a2
B (Dense)
   b1
       b11
       b12
   b2
       b21
       b22
C (Dense)
   c1
   c2
   c3
```

```
D (Sparse)
   d1
   d2
       d21
       d22
E (Sparse)
   e1
   e2
   e3
```

The block in Figure 19 represents the three dense dimensions from within the combination of the sparse members d22 and e3 in the preceding database outline. In Analytic Services, member combinations are denoted by the cross-dimensional operator. The symbol for the cross-dimensional operator is ->. So d22, e3 is written d22 -> e3. A, b21, c3 is written A -> b21 -> c3.

Figure 19: Data Block Representing Dense Dimensions for d22 -> e3



Analytic Services creates a data block for every unique combination of the members of the sparse dimensions D and E (providing that at least one data value exists for the combination).

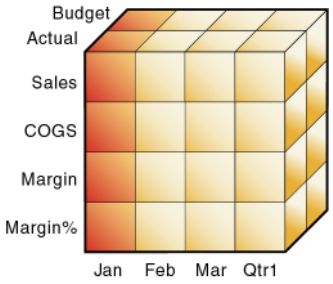
Data blocks, such as the one shown in Figure 19, may include cells that do not contain data values. A data block is created if at least one data value exists in the block. Analytic Services compresses data blocks with missing values on disk, expanding each block fully as it brings the block into memory. Data compression is optional, but is enabled by default. For more information, see "Data Compression" on page 1044.

By carefully selecting dense and sparse standard dimensions, you can ensure that data blocks do not contain many empty cells, minimizing disk storage requirements and improving performance.

Multiple Data Views

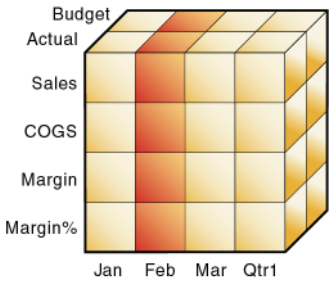
A multidimensional database supports multiple views of data sets for users who need to analyze the relationships between data categories. Slicing the database in different ways gives you different perspectives of the data. The slice of January in Figure 20, for example, examines all data values for which the Year dimension is fixed at Jan.

Figure 20: Data for January



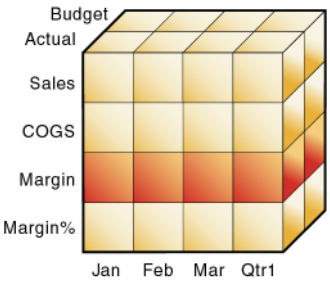
The slice in Figure 21 shows data for the month of February:

Figure 21: Data for February



The slice in Figure 22 shows data for profit margin:

Figure 22: Data for Profit Margin



Chapter 3

Quick Start for Implementing Analytic Services

The table in this chapter provides process steps to help you get up and running with Analytic Services. The table also tells you where you can find more information about each step. Unless otherwise specified, the chapters refer to the *Essbase Analytic Services Database Administrator's Guide*.

Note: The information in this chapter is designed for block storage databases. Some of the information is not relevant to aggregate storage databases. For detailed information on the differences between aggregate and block storage, see Chapter 57, "Comparison of Aggregate and Block Storage."

Note: This chapter assumes that you are a new Analytic Services user. If you are migrating from a previous version of Analytic Services, see the *Essbase Analytic Services Installation Guide* for important migration information.

Table 1: A Process Map

Process Step	Reference
Learn the fundamentals of Analytic Services and distributed OLAP.	• Chapter 1, "Introducing Hyperion Essbase" • Chapter 2, "Understanding Multidimensional Databases" • Chapter 4, "Basic Architectural Elements" • Chapter 7, "Creating Applications and Databases" • Chapter 10, "Working with Attributes" • Chapter 13, "Designing Partitioned Applications" • Attend an Analytic Services training class; contact your software provider for details.
Assess your needs and requirements. Have a clear idea of your data analysis needs and of what types of calculations and reports you want to run.	Your budget, forecasting, and other financial reports with notes on how you want to improve them
Analyze your data from a multidimensional perspective. Consider the following: • Where are your data sources? • What type is the data? Is it detailed, relational data or is it higher-level, hierarchical data that can be used for analysis? • In what format is the data? • How will you access the data? If you need to access relational data, you may need SQL Interface or Integration Services (a separately purchasable product).	• Chapter 2, "Understanding Multidimensional Databases" • *Essbase Analytic Services SQL Interface Guide* • *ODBC drivers documentation* • Essbase Integration Services documentation
Install Analytic Services. Decide what components you want to install. Be aware that the license your company purchased might not include all options.	*Essbase Analytic Services Installation Guide*

Table 1: A Process Map (Continued)

Process Step	Reference
Design your application and database. • Identify business and user requirements, including security. • Identify source data and determine the scope of the Analytic Services database. • Choose whether to leave lowest-level member data in a relational database and access with Hybrid Analysis, or to load all data. • Define standard dimensions and designate sparse and dense storage. • Identify any need for attribute dimensions. • Identify any need for currency conversion applications that track data in different currencies. • Define calculations needed for outline dimensions and members. • Identify any need to monitor data changes in the database. You monitor data changes using the Analytic Services triggers feature (licensed separately).	Chapter 5, "Case Study: Designing a Single-Server, Multidimensional Database"
Estimate the size of your database, check disk space, and ensure that the sizes of the index, data files, and data caches in memory are adequate.	Appendix A, "Limits"
Create an application and a database.	Chapter 7, "Creating Applications and Databases"
Design a currency application.	Chapter 12, "Designing and Building Currency Conversion Applications"
Build an outline for your database.	Chapter 8, "Creating and Changing Database Outlines"
Assign alias names to your members.	Chapter 9, "Setting Dimension and Member Properties"

Table 1: A Process Map (Continued)

Process Step	Reference
Build the dimensions. Decide whether your data loads will introduce new members into the outline. If so, consider dynamically building your dimensions using a rules file and a data source. If not, set up regular data loads.	• Chapter 16, "Understanding Data Loading and Dimension Building" • Chapter 17, "Creating Rules Files"
Load your data. You can load data these ways: • Free-form • With a rules file • With Hybrid Analysis	• Chapter 16, "Understanding Data Loading and Dimension Building" • Chapter 17, "Creating Rules Files" • Chapter 18, "Using a Rules File to Perform Operations on Records, Fields, and Data" • Chapter 19, "Performing and Debugging Data Loads or Dimension Builds" • Chapter 20, "Understanding Advanced Dimension Building Concepts"
Calculate your database. • Decide on a type of calculation: outline or calculation script, or a combination of both. • Ensure that relationships between members and member consolidations in the database outline are correct. • Consider whether tagging some members as Dynamic Calc or whether using Intelligent Calculation will improve calculation efficiency. • Consider which members you need to tag as two-pass calculation to ensure correct calculation results.	• Chapter 21, "Calculating Analytic Services Databases" • Chapter 22, "Developing Formulas" • Chapter 23, "Reviewing Examples of Formulas" • Chapter 24, "Defining Calculation Order" • Chapter 25, "Dynamically Calculating Data Values" • Chapter 26, "Calculating Time Series Data" • Chapter 27, "Developing Calculation Scripts" • Chapter 28, "Reviewing Examples of Calculation Scripts" • Chapter 29, "Developing Custom-Defined Calculation Macros" • Chapter 30, "Developing Custom-Defined Calculation Functions"

Table 1: A Process Map (Continued)

Process Step	Reference
Learn about dynamic calculations and how they can greatly improve performance.	Chapter 25, "Dynamically Calculating Data Values"
View data with Spreadsheet Add-in, other Hyperion tools, or third-party tools.	• For Spreadsheet Add-in, see the *Essbase Spreadsheet Add-in User's Guide* • For other tools, see the documentation for that particular tool • For a list of tools, visit the Hyperion site: `http://www.hyperion.com`
Learn about Partitioning. Think about whether your data can benefit from being decentralized into connected databases.	• Chapter 13, "Designing Partitioned Applications" • Chapter 14, "Creating and Maintaining Partitions"
Link files or cell notes to data cells.	Chapter 11, "Linking Objects to Analytic Services Data"
Copy or export data subsets.	Chapter 34, "Copying Data Subsets and Exporting Data to Other Programs"
Back up and restore your data.	Chapter 47, "Backing Up and Restoring Data"

Table 1: A Process Map (Continued)

Process Step	Reference
Allocate storage and specify Analytic Services kernel settings for your database. • Data compression: Specify data compression on disk and the compression scheme. • Cache sizes: You can specify the index, data file, and data cache sizes. To prevent a slow-down of the operating system, ensure that the sum of index and data cache sizes for all the active databases on the server is not more than two-thirds of the system's RAM. • *Cache memory locking*: You can lock the memory that is used for the index, data file, and data caches into physical memory. • Disk volumes: You can specify the storage location of Analytic Services index files and data files, specify the appropriate disk volume names and configuration parameters. • Isolation level: Specify either committed access or uncommitted access.	• Chapter 44, "Managing Database Settings" • Chapter 45, "Allocating Storage and Compressing Data"
Generate a report. • Choose a type of report: structured or free-form. • Plan the elements of the report, such as page layout, number of columns, identity of members, format of data values, and content of titles. • For a structured report, create page, column, and row headings (unnecessary for a free-form report). • Create and test a report script, use Administration Services' Report Script Editor or any other text editor. • Save the report on OLAP Server or on a client computer.	• Chapter 31, "Understanding Report Script Basics" • Chapter 32, "Developing Report Scripts" • Chapter 34, "Copying Data Subsets and Exporting Data to Other Programs" • Chapter 33, "Mining an Analytic Services Database"

Table 1: A Process Map (Continued)

Process Step	Reference
Fine-tune your database performance and storage settings.	• Chapter 44, "Managing Database Settings" • Chapter 45, "Allocating Storage and Compressing Data" • Chapter 49, "Monitoring Performance"
Automate routine operations by using MaxL or ESSCMD.	• Chapter 48, "Using MaxL Data Definition Language" • Appendix D, "Using ESSCMD"
Design security for your database. • Create a security plan for your environment based on database security needs. • Create users and groups and assign them administrative or data-access permissions, if necessary. • Define common data access permissions at the scope of the server, applications, databases, or data-cell levels. • To define global application or database permissions, select the relevant application or application and database and adjust the settings.	• Chapter 36, "Managing Security for Users and Applications" • Chapter 37, "Controlling Access to Database Cells" • Chapter 38, "Security Examples"
Maintain your applications.	• Chapter 41, "Running Analytic Servers, Applications, and Databases" • Chapter 42, "Managing Applications and Databases" • Chapter 43, "Monitoring Data, Applications, and Databases" • Chapter 44, "Managing Database Settings" • Chapter 45, "Allocating Storage and Compressing Data" • Chapter 46, "Ensuring Data Integrity" • Chapter 47, "Backing Up and Restoring Data"

Table 1: A Process Map (Continued)

Process Step	Reference
Analyze and improve performance and troubleshoot errors if they occur. • Ensure that block size is not excessively large. • Set the correct size for the index, data file, data, and calculator caches. • Validate the database to ensure data integrity. • Consider using partitioning to distribute data across multiple cubes for better scalability and performance. • Ensure that disk space is adequate to allow the application to grow over time. • Archive data from OLAP Server on a regular basis. • Enable logging for spreadsheet update to ensure that log files are updated after archiving. • If sorting on retrievals, increase the size of the retrieval sort buffer.	• Chapter 49, “Monitoring Performance” • Chapter 50, “Improving Analytic Services Performance” • Chapter 51, “Optimizing Analytic Services Caches” • Chapter 52, “Optimizing Database Restructuring” • Chapter 53, “Optimizing Data Loads” • Chapter 54, “Optimizing Calculations” • Chapter 55, “Optimizing with Intelligent Calculation” • Chapter 56, “Optimizing Reports and Other Types of Retrieval”

Chapter 4

Basic Architectural Elements

In this chapter, you will learn how Essbase Analytic Services improves performance by reducing storage space and speeding up data retrieval for multidimensional databases.

Note: The information in this chapter is designed for block storage databases. Some of the information is not relevant to aggregate storage databases. For detailed information on the differences between aggregate and block storage, see Chapter 57, "Comparison of Aggregate and Block Storage."

This chapter contains the following sections:

Attribute Dimensions and Standard Dimensions

Analytic Services has two types of dimensions: attribute dimensions and standard dimensions (non-attribute dimensions). This chapter primarily considers standard dimensions because Analytic Services does not allocate storage for attribute dimension members. Instead it dynamically calculates the members when the user requests data associated with them.

An attribute dimension is a special type of dimension that is associated with a standard dimension. For comprehensive discussion of attribute dimensions, see Chapter 10, "Working with Attributes."

Sparse and Dense Dimensions

Most data sets of multidimensional applications have two characteristics:

- Data is *not* smoothly and uniformly distributed.
- Data does *not* exist for the majority of member combinations. For example, all products may not be sold in all areas of the country.

Analytic Services maximizes performance by dividing the standard dimensions of an application into two types: *dense dimensions* and *sparse dimensions*. This division allows Analytic Services to cope with data that is not smoothly distributed, without losing the advantages of matrix-style access to the data. Analytic Services speeds up data retrieval while minimizing the memory and disk requirements.

Most multidimensional databases are inherently sparse: they lack data values for the majority of member combinations. A sparse dimension is a dimension with a low percentage of available data positions filled.

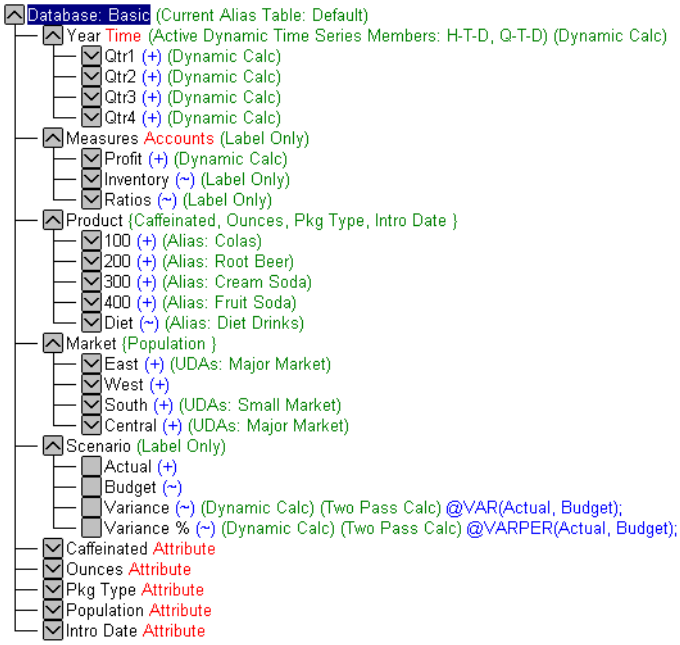
For example, the Sample Basic database shown in Figure 23 includes the Year, Product, Market, Measures, and Scenario dimensions. Product represents the product units, Market represents the geographical regions in which the products are sold, and Measures represents the accounts data. Because not every product is sold in every market, Market and Product are chosen as sparse dimensions.

Most multidimensional databases also contain dense dimensions. A dense dimension is a dimension with a high probability that one or more data points is occupied in every combination of dimensions. For example, in the Sample Basic database, accounts data exists for almost all products in all markets, so Measures

is chosen as a dense dimension. Year and Scenario are also chosen as dense dimensions. Year represents time in months, and Scenario represents whether the accounts values are budget or actual values.

Note: Caffeinated, Intro Date, Ounces, and Pkg Type are attribute dimensions that are associated with the Product dimension. Population is an attribute dimension that is associated with the Market dimension. Members of attribute dimensions describe characteristics of the members of the dimensions with which they are associated. For example, each product has a size in ounces. Attribute dimensions are always sparse dimensions and must be associated with a sparse standard dimension. Analytic Services does not store the data for attribute dimensions, Analytic Services dynamically calculates the data when a user retrieves it. For a comprehensive discussion about attribute dimensions, see Chapter 10, "Working with Attributes."

Figure 23: Sample Basic Database Outline



Data Blocks and the Index System

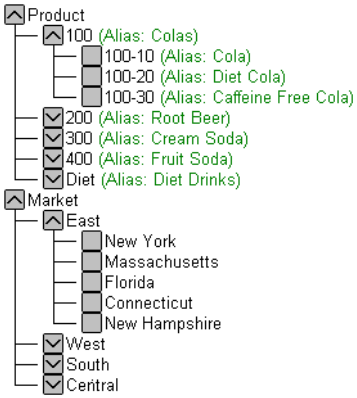
Analytic Services uses two types of internal structures to store and access data: data blocks and the index system.

Analytic Services creates a *data block* for each unique combination of sparse standard dimension members (providing that at least one data value exists for the sparse dimension member combination). The data block represents all the dense dimension members for its combination of sparse dimension members.

Analytic Services creates an *index entry* for each data block. The index represents the combinations of sparse standard dimension members. It contains an entry for each unique combination of sparse standard dimension members for which at least one data value exists.

For example, in the Sample Basic database outline shown in Figure 24, Product and Market are sparse dimensions.

Figure 24: Product and Market Dimensions from the Sample Basic Database



If data exists for Caffeine Free Cola in New York, then Analytic Services creates a data block and an index entry for the sparse member combination of Caffeine Free Cola (100-30) -> New York. If Caffeine Free Cola is *not* sold in Florida, then Analytic Services does *not* create a data block or an index entry for the sparse member combination of Caffeine Free Cola (100-30) -> Florida.

The data block Caffeine Free Cola (100-30) -> New York represents all the Year, Measures, and Scenario dimensions for Caffeine Free Cola (100-30) -> New York.

Each unique data value can be considered to exist in a cell in a data block. When Analytic Services searches for a data value, it uses the index to locate the appropriate data block as shown in Figure 25. Then, within the data block, it locates the cell containing the data value. The index entry provides a pointer to the data block. The index handles sparse data efficiently because it includes only pointers to existing data blocks.

Figure 25: Simplified Index and Data Blocks

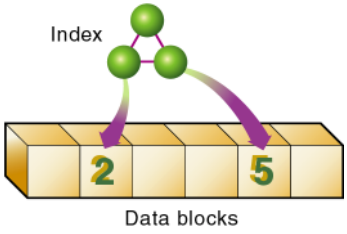
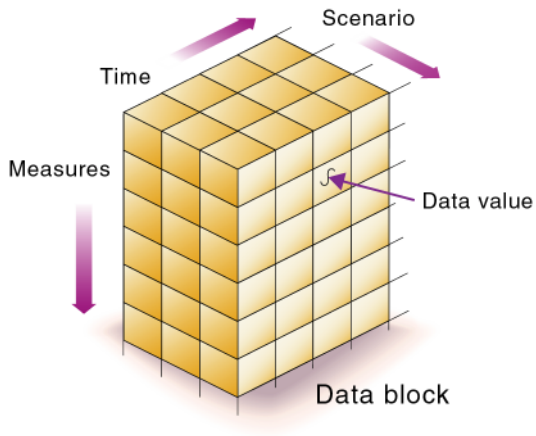


Figure 26 shows part of a data block for the Sample Basic database. Each dimension of the block represents a dense dimension in the Sample Basic database: Time, Measures, and Scenario. A data block exists for each unique combination of members of the Product and Market sparse dimensions (providing that at least one data value exists for the combination).

Figure 26: Part of a Data Block for the Sample Basic Database



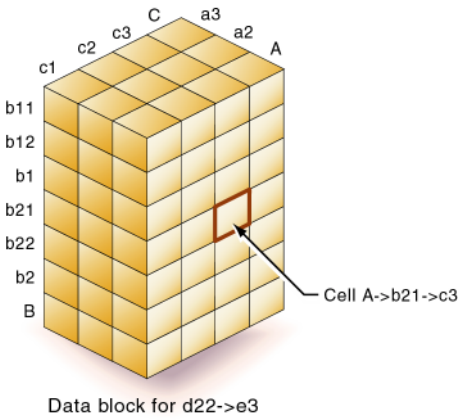
Each data block is a multidimensional array that contains a fixed, ordered location for each possible combination of dense dimension members. Accessing a cell in the block does not involve sequential or index searches. The search is almost instantaneous, resulting in optimal retrieval and calculation speed.

Analytic Services orders the cells in a data block according to the order of the members in the dense dimensions of the database outline.

```
A (Dense)
   a1
   a2
B (Dense)
   b1
       b11
       b12
   b2
       b21
       b22
C (Dense)
   c1
   c2
   c3
D (Sparse)
   d1
   d2
       d21
       d22
E (Sparse)
   e1
   e2
   e3
```

The block in Figure 27 represents the three dense dimensions from within the combination of the sparse members d22 and e3 in the preceding database outline. In Analytic Services, member combinations are denoted by the cross-dimensional operator. The symbol for the cross-dimensional operator is ->. So d22, e3 is written d22 -> e3. A, b21, c3 is written A -> b21 -> c3.

Figure 27: Data Block Representing Dense Dimensions for d22 -> e3



Analytic Services creates a data block for every unique combination of the members of the sparse dimensions D and E (providing that at least one data value exists for the combination).

Data blocks, such as the one shown in Figure 27, may include cells that do not contain data values. A data block is created if at least one data value exists in the block. Analytic Services compresses data blocks with missing values on disk, expanding each block fully as it brings the block into memory. Data compression is optional, but is enabled by default. For more information, see “Data Compression” on page 1044.

By carefully selecting dense and sparse standard dimensions, you can ensure that data blocks do not contain many empty cells. In Analytic Services, empty cells are known as missing or #MISSING data. You can also minimize disk storage requirements and maximize performance.

Selection of Sparse and Dense Dimensions

In most data sets, existing data tends to follow predictable patterns of density and sparsity. If you match patterns correctly, you can store the existing data in a reasonable number of fairly dense data blocks, rather than in many highly sparse data blocks.

When you add a dimension to an outline in Outline Editor, Analytic Services automatically sets the dimension as sparse. To help you determine whether dimensions should be dense or sparse, Analytic Services provides an automatic configuration feature.

➤ To select automatic configuration of dense and sparse dimensions use the following method:

Tool	Topic	Location
Administration Services	Outline Editor - Properties Tab, Auto configure option Setting Dimensions as Dense or Sparse	*Essbase Administration Services Online Help*

If you select automatic configuration, you cannot manually set the sparse or dense property for each dimension. Turn off automatic configuration to set the sparse and dense property manually. Attribute dimensions are always sparse dimensions. Keep in mind that you can associate attribute dimensions only with sparse standard dimensions.

Note: The automatic configuration of dense and sparse dimensions provides only an estimate. It cannot take into account the nature of the data you will load into your database or multiple user considerations.

Determining the Sparse-Dense Configuration for Sample Basic

Consider the Sample Basic database that is shipped with Analytic Services. The Sample Basic database represents data for The Beverage Company (TBC).

TBC does not sell every product in every market; therefore, the data set is reasonably sparse. Data values do not exist for many combinations of members in the Product and Market dimensions. For example, if Caffeine Free Cola is not sold

in Florida, then data values do not exist for the combination Caffeine Free Cola (100-30)->Florida. So, Product and Market are sparse dimensions. Therefore, if no data values exist for a specific combination of members in these dimensions, Analytic Services does not create a data block for the combination.

However, consider combinations of members in the Year, Measures, and Scenario dimensions. Data values almost always exist for some member combinations on these dimensions. For example, data values exist for the member combination Sales->January->Actual because at least some products are sold in January. Thus, Year and, similarly, Measures and Scenario are dense dimensions.

The sparse-dense configuration of the standard dimensions in the Sample Basic database may be summarized as follows:

- The sparse standard dimension are Product and Market.
- The dense standard dimensions are Year, Measures, and Scenario.

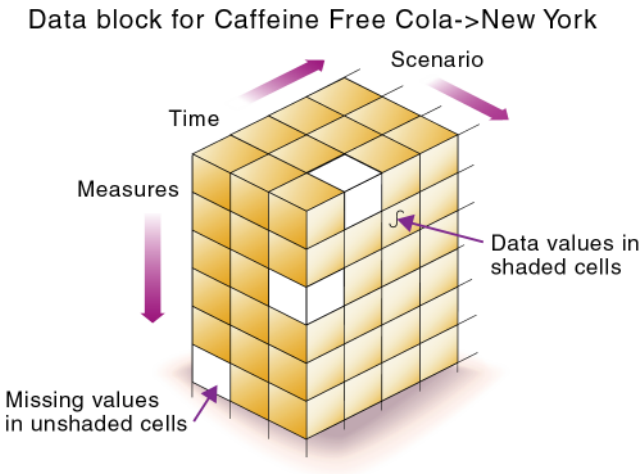
Analytic Services creates a data block for each unique combination of members in the Product and Market dimensions. Each data block represents data from the dense dimensions. The data blocks are likely to have few empty cells.

For example, consider the sparse member combination Caffeine Free Cola (100-30), New York, illustrated by Figure 28:

- If accounts data (represented by the Measures dimension) exists for this combination for January, it probably exists for February and for all members in the Year dimension.
- If a data value exists for one member on the Measures dimension, then it is likely that other accounts data values exist for other members in the Measures dimension.

- If Actual accounts data values exist, then it is likely that Budget accounts data values exist.

Figure 28: Dense Data Block for Sample Basic Database



Dense and Sparse Selection Scenarios

The following scenarios show how a database is affected when you select different dense and sparse standard dimensions. Assume that these scenarios are based on typical databases with at least seven dimensions and several hundred members:

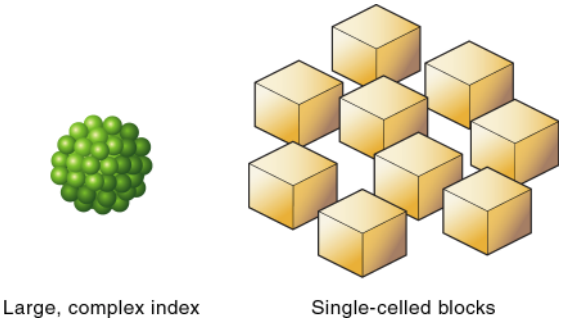
- “Scenario 1: All Sparse Standard Dimensions” on page 70
- “Scenario 2: All Dense Standard Dimensions” on page 71
- “Scenario 3: Dense and Sparse Standard Dimensions” on page 72
- “Scenario 4: A Typical Multidimensional Problem” on page 72

Scenario 1: All Sparse Standard Dimensions

If you make all dimensions sparse, Analytic Services creates data blocks that consist of single *data cells* that contain single data values. An index entry is created for each data block and, therefore, in this scenario, for each existing data value.

This configuration produces a huge index that requires a large amount of memory. The more index entries, the longer Analytic Services searches to find a specific block.

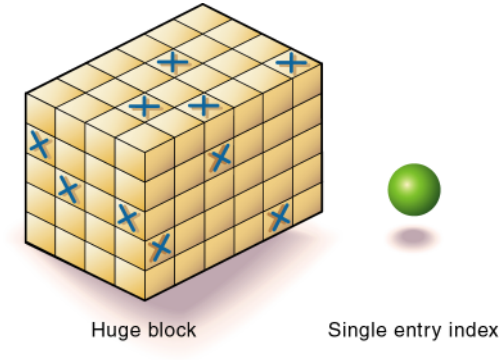
Figure 29: Database with All Sparse Standard Dimensions



Scenario 2: All Dense Standard Dimensions

If you make all dimensions dense, as shown in Figure 30, Analytic Services creates one index entry and one very large, very sparse block. In most applications, this configuration requires thousands of times more storage than other configurations. Analytic Services needs to load the entire block into memory when it searches for a data value, which requires enormous amounts of memory.

Figure 30: Database with All Dense Standard Dimensions

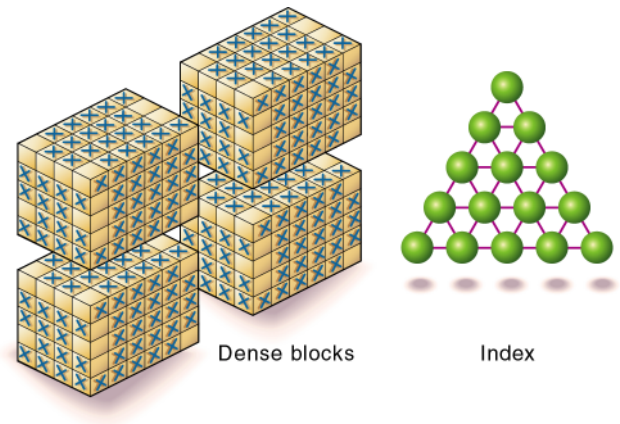


Scenario 3: Dense and Sparse Standard Dimensions

Based upon your knowledge of your company's data, you have identified all your sparse and dense standard dimensions. Ideally, you have approximately equal numbers of sparse and dense standard dimensions. If not, you are probably working with a non-typical data set and you need to do more tuning to define the dimensions.

Analytic Services creates dense blocks that can fit into memory easily and creates a relatively small index as shown in Figure 31. Your database runs efficiently using minimal resources.

Figure 31: An Ideal Configuration with Combination of Dense and Sparse Dimensions

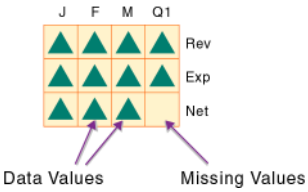


Scenario 4: A Typical Multidimensional Problem

Consider a database with four standard dimensions: Time, Accounts, Region, and Product. In the following example, Time and Accounts are dense dimensions, and Region and Product are sparse dimensions.

The two-dimensional data blocks shown in Figure 32 represent data values from the dense dimensions: Time and Accounts. The members in the Time dimension are J, F, M, and Q1. The members in the Accounts dimension are Rev, Exp, and Net.

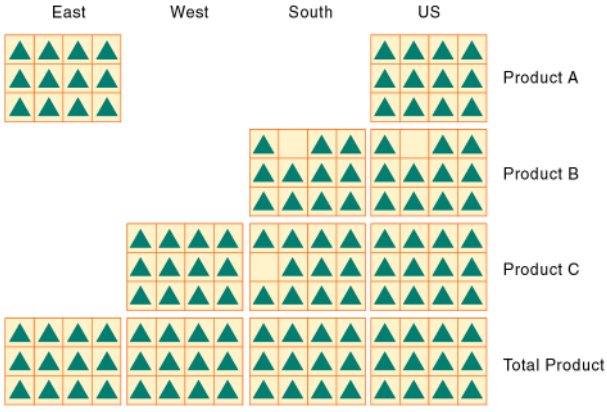
Figure 32: Two-dimensional Data Block for Time and Accounts



Analytic Services creates data blocks for combinations of members in the sparse standard dimensions (providing at least one data value exists for the member combination). The sparse dimensions are Region and Product. The members of the Region dimension are East, West, South, and Total US. The members in the Product dimension are Product A, Product B, Product C, and Total Product.

Figure 33 shows 11 data blocks. No data values exist for Product A in the West and South, for Product B in the East and West, and for Product C in the East. Therefore, Analytic Services has not created data blocks for these member combinations. The data blocks that Analytic Services has created have very few empty cells.

Figure 33: Data Blocks Created for Sparse Members on Region and Product

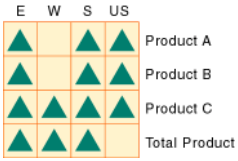


This example effectively concentrates all the sparseness into the index and concentrates all the data into fully utilized blocks. This configuration provides efficient data storage and retrieval.

Now consider a reversal of the dense and sparse dimension selections. In the following example, Region and Product are dense dimensions, and Time and Accounts are sparse dimensions.

As shown in Figure 34, the two-dimensional data blocks represent data values from the dense dimensions: Region and Product.

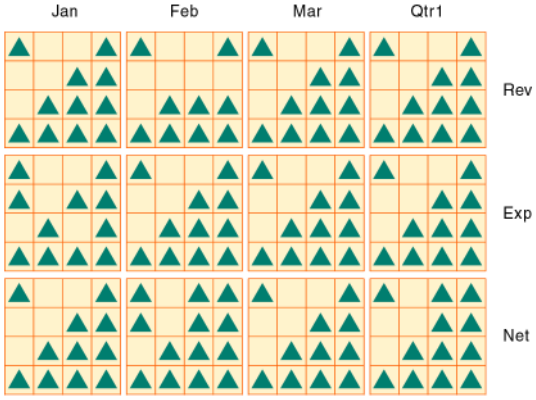
Figure 34: Two-Dimensional Data Block for Region and Product



Analytic Services creates data blocks for combinations of members in the sparse standard dimensions (providing at least one data value exists for the member combination). The sparse standard dimensions are Time and Accounts.

Figure 35 shows 12 data blocks. Data values exist for all combinations of members in the Time and Accounts dimensions; therefore, Analytic Services creates data blocks for all the member combinations. Because data values do not exist for all products in all regions, the data blocks have many empty cells. Data blocks with many empty cells store data inefficiently.

Figure 35: Data Blocks Created for Sparse Members on Time and Accounts



The Analytic Services Solution

When you create an optimized Analytic Services database, you need to consider carefully the following questions:

- How does your company use the data?
- How do you plan to build and order the dimensions?
- Which data compression scheme will you use?
- How do you want to create and order calculations?

For more information on:

- Planning the development of your multidimensional database, see Chapter 5, "Case Study: Designing a Single-Server, Multidimensional Database."
- Selecting dense and sparse dimensions, see "Sparse and Dense Dimensions" on page 62.
- Loading data, see Chapter 16, "Understanding Data Loading and Dimension Building."
- Compressing data and optimizing your database, see "Data Compression" on page 1044.
- Calculating your database, see Chapter 21, "Calculating Analytic Services Databases."

Chapter 5

Case Study: Designing a Single-Server, Multidimensional Database

To implement a multidimensional database, first you install Essbase Analytic Services, and then you design and create an application and databases. You analyze data sources and define requirements very carefully and then decide whether a single-server approach or a partitioned, distributed approach best serves your needs. For criteria that you can review to decide whether to partition an application, see "Deciding Whether to Partition a Database" on page 239.

Using a case study, this chapter provides an overview of the database planning process and discusses working rules that you can follow to design a single-server, multidimensional database solution for your organization. For detailed information about building applications and databases, see Chapter 7, "Creating Applications and Databases."

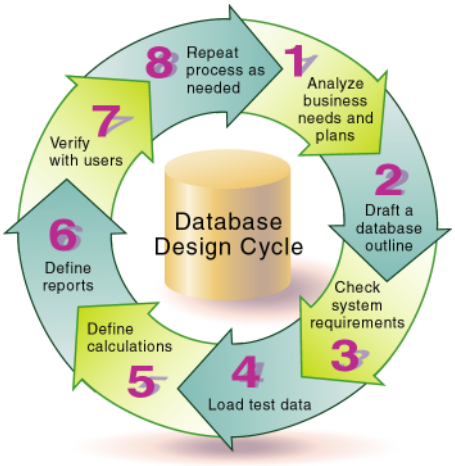
Note: The information in this chapter is designed for block storage databases. Some of the information is not relevant to aggregate storage databases. For detailed information on the differences between aggregate and block storage, see Chapter 57, "Comparison of Aggregate and Block Storage."

This chapter includes the following topics:

Process for Designing a Database

As illustrated in Figure 36, designing an application is a cyclic process that moves from a planning stage to a verification stage.

Figure 36: The Database Design Cycle



The database design process includes the following basic steps:

1. Analyze business needs and design a plan.

   The application and database that you create must satisfy the information needs of your users and your organization. Therefore, you identify source data, define user information access needs, review security considerations, and design a database model. See "Analyzing and Planning" on page 80.

2. Draft a database outline.

   The *outline* determines the structure of the database—what information is stored and how different pieces of information relate to one another. See "Drafting Outlines" on page 95.

3. Check system requirements.

   How you meet system requirements and define system parameters affects the efficiency and performance of the database. See "Checking System Requirements" on page 102.

4. Load test data into the database.

   After an outline and a security plan are in place, you load the database with test data to enable the later steps of the process. See "Loading Test Data" on page 103.

5. Define calculations.

   You test outline consolidations and write and test formulas and calculation scripts for specialized calculations. See "Defining Calculations" on page 103.

6. Define reports.

   Users access data through print and online reports and spreadsheets or on the World Wide Web. If you plan to provide predefined reports to users, you design report layouts and run reports. See "Defining Reports" on page 115.

7. Verify with users.

   You want to ensure that the database satisfies your user goals. You must solicit and carefully consider the opinions of users. See "Verifying the Design" on page 116.

8. Repeat the process.

   To fine-tune the design, you repeat steps 1 through 7.

Case Study: The Beverage Company

This chapter bases the database planning process on the needs of a fictitious company called *The Beverage Company* (TBC) and uses TBC as an example to demonstrate how to build an Analytic Services database. The examples follow a variation of the Sample Basic application that is included with the Analytic Services installation.

TBC manufactures, markets, and distributes soft drink products internationally. Analysts at TBC prepare budget forecasts and compare performance to budget forecasts on a monthly basis. The financial measures that analysts track are profit and loss and inventory.

TBC uses spreadsheet packages to prepare budget data and perform variance reporting. Because TBC plans and tracks a variety of products over several markets, the process of deriving and analyzing data is tedious. Last month, analysts spent most of their time entering and rekeying data and preparing reports.

TBC has determined that Analytic Services is the best tool for creating a centralized repository for financial data. The data repository will reside on a server that is accessible to analysts throughout the organization. Users will have access to the server and will be able to load data from various sources and retrieve data as needed. TBC has a variety of users, so TBC expects that different users will have different security levels for accessing data.

Analyzing and Planning

The design and operation of an Analytic Services multidimensional database plays a key role in achieving a well-tuned system that enables you to analyze business information efficiently. Given the size and performance volatility of multidimensional databases, developing an optimized database is critical. A detailed plan that outlines data sources, user needs, and prospective database elements can save you development and implementation time.

The planning and analysis phase involves three tasks:

- "Analyzing Source Data" on page 81
- "Identifying User Requirements" on page 82
- "Planning for Security in a Multiple User Environment" on page 82
- "Creating Database Models" on page 82

When designing a multidimensional application, consider these factors:

- How information flows within the company—who uses what data for what purposes
- The types of reporting the company does—what types of data must be included in the outline to serve user reporting needs

Note: The best practices recommendation is to define only one database per application. There are several reasons for this recommendation, including enhanced memory usage and ease of database administration. Applications that use the optional Analytic Services currency conversion module are an exception to this recommendation. Currency conversion applications generally consist of a main database and a separate currency database (see Chapter 12, "Designing and Building Currency Conversion Applications").

Analyzing Source Data

First, you need to evaluate the source data that you want to include in the database. Think about where the data resides and how often you plan to update the database with the data. This up-front research saves time when you create the database outline and load data into the Analytic Services database.

Determine the scope of the database. If an organization has thousands of product families containing hundreds of thousands of products, you may want to store data values only for product families. Interview members from each user department to find out what data they process, how they process data today, and how they want to process data in the future.

Carefully define reporting and analysis needs.

- How do users want to view and analyze data?
- How much detail should the database contain?
- Does the data support the desired analysis and reporting goals?
- If not, what additional data do you need and where can you find the needed data?

Determine the location of the current data.

- Where does each department currently store data?
- Is data in a form that Analytic Services can use?
- Do departments store data in a DB2 database on an IBM mainframe, in a relational database on a UNIX-based server, or in a PC-based database or spreadsheet?
- Who updates the database and how frequently?
- Do the individuals who need to update data have access to the data?

Make sure that the data is ready to load into Analytic Services.

- Does data come from a single source or from multiple sources?
- Is data in a format that Analytic Services can import? For a list of valid data sources that you can import into Analytic Services, see "Data Sources" on page 356.
- Is all data that you want to use readily available?

Identifying User Requirements

Be sure to discuss information needs with users. Review the information they use and the reports they must generate for review by others. Determine the following requirements.

- What types of analysis do users require?
- What summary and detail levels of information do users need?
- Do some users require access to information that other users should not see?

Planning for Security in a Multiple User Environment

The time to think about the type of security permissions you plan to issue for an Analytic Services database is when you consider user information needs. End your analysis with a list of users and permissions.

Use this checklist to plan for security:

- Who are the users and what permissions should they have?
- Who should have load data permissions?
- Which users can be grouped, and as a group, given similar permissions?

See Chapter 36, “Managing Security for Users and Applications” for information about assigning user permissions.

Creating Database Models

You are now ready to create a model of the database on paper. To build the model, identify the perspectives and views that are important to your business. These views translate into the dimensions of the database model.

Most businesses choose to analyze the following areas:

- Time periods
- Accounting measures
- Scenarios
- Products

- Distribution channels
- Geographical regions
- Business units

Use the following topics to help you gather information and make decisions:

- "Identifying Analysis Objectives" on page 83
- "Determining Dimensions and Members" on page 83
- "Analyzing Database Design" on page 88

Identifying Analysis Objectives

After you identify the major areas of information in a business, the next step in designing an Analytic Services database is deciding how the database enables data analysis:

- If analyzing by time, which time periods are needed? Does the analysis need to include only the current year or multiple years? Does the analysis need to include quarterly and monthly data? Does the analysis need to include data by season?
- If analyzing by geographical region, how do you define the regions? Do you define regions by sales territories? Do you define regions by geographical boundaries such as states and cities?
- If analyzing by product line, do you need to review data for each specific product? Can you summarize data into product classes?

Regardless of the business area, you need to determine the perspective and detail needed in the analysis. Each business area that you analyze provides a different view of the data.

Determining Dimensions and Members

You can represent each of the business views as a separate standard dimension in the database. If you need to analyze a business area by classification or attribute, such as by the size or color of products, you can use attribute dimensions to represent the classification views.

The dimensions that you choose determine what types of analysis you can perform on the data. With Analytic Services, you can use as many dimensions as you need for analysis. A typical Analytic Services database contains at least seven standard dimensions (non-attribute dimensions) and many more attribute dimensions.

When you have an idea of what dimensions and members you need, review the following topics and develop a tentative database design:

- "Relationships Among Dimensions" on page 84
- "Example Dimension-Member Structure" on page 85
- "Checklist for Determining Dimensions and Members" on page 87

After you determine the dimensions of the database model, choose the elements or items within the perspective of each dimension. These elements become the members of their respective dimensions. For example, a perspective of time may include the time periods that you want to analyze, such as quarters, and within quarters, months. Each quarter and month becomes a member of the dimension that you create for time. Quarters and months represent a two-level hierarchy of members and their children. Months within a quarter consolidate to a total for each quarter.

Relationships Among Dimensions

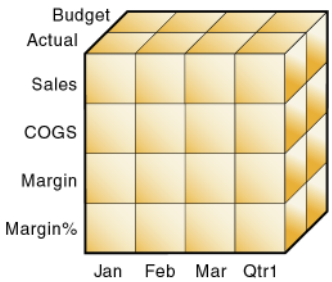
Next, consider the relationships among the business areas. The structure of an Analytic Services database makes it easy for users to analyze information from many different perspectives. A financial analyst, for example, may ask the following questions:

- What are sales for a particular month? How does this figure compare to sales in the same month over the last five years?
- By what percentage is profit margin increasing?
- How close are actual values to budgeted values?

In other words, the analyst may want to examine information from three different perspectives—time, account, and scenario. The sample database shown in Figure 37 represents these three perspectives as three dimensions, with one dimension represented along each of the three axes:

- A time dimension, which consists of the individual months Jan, Feb, and Mar and the total for Qtr1, is displayed along the X-axis.
- An accounts dimension, which consists of accounting figures such as Sales, COGS, Margin, and Margin%, is displayed along the Y-axis.
- Another dimension which provides a different point of view, such as Budget for budget values and Actual for actual values, is displayed along the Z-axis.

Figure 37: Cube Representing Three Database Dimensions



The cells within the cube, where the members intersect, contain the data relevant to all three intersecting members; for example, the actual sales in January.

Example Dimension-Member Structure

Table 2 shows a summary of the TBC business areas that the planner determined would be dimensions. The dimensions represent the major business areas to be analyzed. The planner created three columns, with the dimensions in the left column and members in the two right columns. The members in column 3 are

subcategories of the members in column 2. In some cases, members in column 3 are divided into another level of subcategories; for example, the Margin of the Measures dimension is divided into Sales and COGS.

Table 2: TBC Sample Dimensions

Dimensions	Members	Child Members
Year	Qtr1	Jan, Feb, Mar
	Qtr2	Apr, May, Jun
	Qtr3	Jul, Aug, Sep
	Qtr4	Oct, Nov, Dec
Measures	Profit	Margin: Sales, COGS Total Expenses: Marketing, Payroll, Miscellaneous
	Inventory	Opening Inventory, Additions, Ending Inventory
	Ratios	Margin %, Profit %, Profit per Ounce
Product	Colas (100)	Cola (100-10), Diet Cola (100-20), Caffeine Free Cola (100-30)
	Root Beer (200)	Old Fashioned (200-10), Diet Root Beer (200-20), Sarsaparilla (200-30), Birch Beer (200-40)
	Cream Soda (300)	Dark Cream (300-10), Vanilla Cream (300-20), Diet Cream Soda (300-30)
	Fruit Soda (400)	Grape (400-10), Orange (400-20), Strawberry (400-30)
Market	East	Connecticut, Florida, Massachusetts, New Hampshire, New York
	West	California, Nevada, Oregon, Utah, Washington
	South	Louisiana, New Mexico, Oklahoma, Texas
	Central	Colorado, Illinois, Iowa, Missouri, Ohio, Wisconsin

Table 2: TBC Sample Dimensions (Continued)

Dimensions	Members	Child Members
Scenario	Actual	
	Budget	
	Variance	
	Variance %	

In addition the planner added two attribute dimensions to enable product analysis based on size and packaging:

Table 3: TBC Sample Attribute Dimensions

Dimensions	Members	Child Members
Ounces	Large Small	64, 32, 20 16, 12
Pkg Type	Bottle Can	

Checklist for Determining Dimensions and Members

Use the following checklist when determining the dimensions and members of your model database:

- What are the candidates for dimensions?
- Do any of the dimensions classify or describe other dimensions? These dimensions are candidates for attribute dimensions.
- Do users want to qualify their view of a dimension? The categories by which they qualify a dimension are candidates for attribute dimensions.
- What are the candidates for members?
- How many levels does the data require?
- How does the data consolidate?

Analyzing Database Design

While the initial dimension design is still on paper, you should review the design according to a set of guidelines. The guidelines help you to fine-tune the database and leverage the multidimensional technology. The guidelines are processes or questions that help you achieve an efficient design and meet consolidation and calculation goals.

Keep in mind that the number of members needed to describe a potential data point should determine the number of dimensions. As you analyze the design, if you are not sure that you should delete a dimension, keep it and apply more analysis rules until you feel confident about deleting or keeping it.

Use the information in the following topics to analyze and, as needed, to improve your database design:

- "Dense and Sparse Dimensions" on page 88
- "Standard and Attribute Dimensions" on page 88
- "Dimension Combinations" on page 90
- "Repetition in Outlines" on page 92
- "Interdimensional Irrelevance" on page 93
- "Reasons to Split Databases" on page 94
- "Checklist to Analyze the Database Design" on page 95

Dense and Sparse Dimensions

You need to decide which dimensions are sparse and which dense. These decisions affect performance. For a basic introduction, see "Sparse and Dense Dimensions" on page 62. For a comprehensive discussion of storage and performance, see "Designing an Outline to Optimize Performance" on page 100.

Standard and Attribute Dimensions

For simplicity, the examples in this topic show alternative arrangements for what was initially designed as two dimensions. You can apply the same logic to all combinations of dimensions.

Consider the design for a company that sells products to multiple customers over multiple markets; the markets are unique to each customer:

```
             Cust A  Cust B  Cust C

New York     100     N/A     N/A
Illinois     N/A     150     N/A
California   N/A     N/A     30
```

Cust A is only in New York, Cust B is only in Illinois, and Cust C is only in California. The company can define the data in one standard dimension:

```
Market
       New York
              Cust A
       Illinois
              Cust B
       California
              Cust C
```

However, if you look at a larger sampling of data, you may see that there can be many customers in each market. Cust A and Cust E are in New York; Cust B, Cust M, and Cust P are in Illinois; Cust C and Cust F are in California. In this situation, the company typically defines the large dimension, Customer, as a standard dimension and the smaller dimension, Market, as an attribute dimension. The company associates the members of the Market dimension as attributes of the members of the Customer dimension. The members of the Market dimension describe locations of the customers.

```
Customer (Standard dimension)
       Cust A   (Attribute:New York)
       Cust B   (Attribute:Illinois)
       Cust C   (Attribute:California)
       Cust E   (Attribute:New York)
       Cust F   (Attribute:California)
       Cust M   (Attribute:Illinois)
       Cust P   (Attribute:Illinois)
Market (Attribute dimension)
       New York
       Illinois
       California
```

Consider another situation. Again, the company sells products to multiple customers over multiple markets. This time, the company can ship to a customer that has locations in different markets:

```
             Cust A  Cust B  Cust C

New York     100      75     N/A
Illinois     N/A     150     N/A
California   150     N/A      30
```

Cust A is in New York and California. Cust B is in New York and Illinois. Cust C is only in California. Using an attribute dimension does not work in this situation; a customer member cannot have more than one attribute member. Therefore, the company designs the data in two standard dimensions:

```
Customer
       Cust A
       Cust B
       Cust C
Market
       New York
       Illinois
       California
```

Dimension Combinations

Break each combination of two dimensions into a two-dimensional matrix. For example, proposed dimensions at TBC (as listed in Table 2 on page 86) include the following combinations:

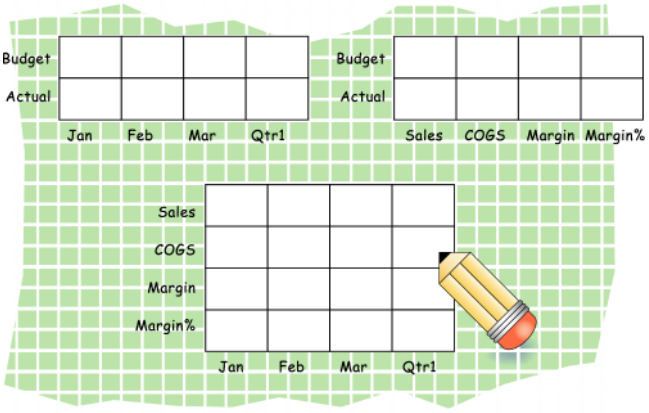
- Year across Measures
- Year across Product
- Year across Market
- Year across Scenario
- Measures across Product
- Measures across Market
- Measures across Scenario
- Market across Product

- Market across Scenario
- Scenario across Product
- Ounces across Pkg Type

As attribute dimensions associated with the Product dimension, Ounces and Pkg Type can be considered with the Product dimension.

To help visualize each dimension, you can draw a matrix and include a few of the first generation members. Figure 38 shows a simplified set of matrixes for three dimensions.

Figure 38: Analyzing Dimensional Relationships



For each combination of dimensions, ask three questions:

- Does it add analytic value?
- Does it add utility for reporting?
- Does it avoid an excess of unused combinations?

For each combination, the answers to the questions help determine if the combination is valid for the database. Ideally, the answers to all questions should be yes. If all answers are not yes, you should consider rearranging the data into dimensions that are more meaningful. As you work through this process, be sure to discuss information needs with users.

Repetition in Outlines

The repetition of elements in an outline often indicates a need to split dimensions. Here is an example of repetition and a solution:

Repetition

```
Accounts
  Budget
    Profit
      Margin
        Sales
        COGS
      Expenses
  Actual
    Profit
      Margin
        Sales
        COGS
      Expenses
```

No Repetition

```
Accounts
  Profit
    Margin
      Sales
      COGS
    Expenses
Scenario
  Budget
  Actual
```

Separating Budget and Actual and placing them into another dimension simplifies the outline and provides a simpler view of the budget and actual figures of the other dimensions in the database.

The left column of this table uses shared members in the Diet dimension to analyze diet beverages. You can avoid the repetition of the left column and simplify the design of the outline by creating a Diet attribute dimension, as shown in the second example.

Repetition

```
Product
   100 (Alias: Colas)
      100-10 (Alias: Cola)
      100-20 (Alias: Diet Cola)
   200 (Alias: Root Beer)
      200-20 (Alias: Diet Root Beer)
      200-30 (Alias: Birch Beer)
   300 (Alias Cream Soda)
      300-10 (Alias: Dark Cream)
      300-20 (Alias: Diet Cream)
   Diet (Alias: Diet Drinks)
      100-20 (Alias: Diet Cola)
      200-20 (Alias: Diet Root Beer)
      300-20 (Alias: Diet Cream)
```

No Repetition

```
Product (Diet)
   100 (Alias: Colas)
      100-10 (Alias: Cola) (Diet: False)
      100-20 (Alias: Diet Cola) (Diet: True)
   200 (Alias: Root Beer)
      200-20 (Alias: Diet Root Beer) (Diet: True)
      200-30 (Alias: Birch Beer) (Diet: False)
   300 (Alias Cream Soda)
      300-10 (Alias: Dark Cream) (Diet: False)
      300-20 (Alias: Diet Cream) (Diet: True)
Diet Attribute (Type: Boolean)
    True
    False
```


Attribute dimensions also provide additional analytic capabilities. For a review of the advantages of using attribute dimensions, see “Designing Attribute Dimensions” on page 192.

Interdimensional Irrelevance

Interdimensional irrelevance occurs when many members of a dimension are irrelevant across other dimensions. Analytic Services defines irrelevant data as data that Analytic Services stores only at the summary (dimension) level. In such a situation, you may be able to remove a dimension from the database and add its members to another dimension or split the model into separate databases.

For example, TBC considered analyzing salaries as a member of the Measures dimension. But salary information often proves to be irrelevant in the context of a corporate database. Most salaries are confidential and apply to specific individuals. The individual and the salary typically represent one cell, with no reason to intersect with any other dimension.

TBC considered separating employees into a separate dimension. Table 4 shows an example of how TBC analyzed the proposed Employee dimension for interdimensional irrelevance. Members of the proposed Employee dimension are compared with members of the Measures dimension. Only the Salary measure is relevant to individual employees.

Table 4: Interdimensional Irrelevance Example

	Joe Smith	Mary Jones	Mike Garcia	All Employees
Revenue				x
Variable Costs				x
COGS				x
Advertising				x
Salaries	x	x	x	x
Fixed Costs				x
Expenses				x
Profit				x

Reasons to Split Databases

As discussed in the previous topic, “Interdimensional Irrelevance” on page 93, TBC agreed that, in context with other dimensions, individual employees were irrelevant. They also agreed that adding an Employee dimension substantially increased database storage needs. Consequently, they decided to create a separate Human Resources (HR) database. The new HR database contains a group of related dimensions and includes salaries, benefits, insurance, and 401(k) plans.

There are many reasons for splitting a database; for example, suppose that a company maintains an organizational database that contains several international subsidiaries located in several time zones. Each subsidiary relies on time-sensitive financial calculations. You may want to split the database for groups of subsidiaries in the same time zone to ensure that financial calculations are timely. You can also use a partitioned application to separate information by subsidiary.

Checklist to Analyze the Database Design

Use the following checklist to analyze the database design:

- Have you minimized the number of dimensions?
- For each dimensional combination, did you ask:
  - Does it add analytic value?
  - Does it add utility for reporting?
  - Does it avoid an excess of unused combinations?
- Did you avoid repetition in the outline?
- Did you avoid interdimensional irrelevance?
- Did you split the databases as necessary?

Drafting Outlines

At this point, you can create the application and database and build the first draft of the outline in Analytic Services. The draft defines all dimensions, members, and consolidations. Use the outline to design consolidation requirements and identify where you need formulas and calculation scripts.

Note: Before you create a database and build its outline, you must create an Analytic Services application to contain it.

The TBC planners issued the following draft for a database outline. In this plan, the bold words are the dimensions—Year, Measures, Product, Market, Scenario, Pkg Type, and Ounces. Observe how TBC anticipated consolidations, calculations and formulas, and reporting requirements. The planners also used product codes rather than product names to describe products.

- **Year.** TBC needs to collect data monthly and summarize the monthly data by quarter and year. Monthly data, stored in members such as Jan, Feb, and Mar, consolidates to quarters. Quarterly data, stored in members such as Qtr1 and Qtr2, consolidates to Year.
- **Measures.** Sales, Cost of Goods Sold, Marketing, Payroll, Miscellaneous, Opening Inventory, Additions, and Ending Inventory are standard measures. Analytic Services can calculate Margin, Total Expenses, Profit, Total Inventory, Profit %, Margin %, and Profit per Ounce from these measures. TBC needs to calculate Measures on a monthly, quarterly, and yearly basis.

- **Product.** The Product codes are 100-10, 100-20, 100-30, 200-10, 200-20, 200-30, 200-40, 300-10, 300-20, 300-30, 400-10, 400-20, and 400-30. Each product consolidates to its respective family (100, 200, 300, and 400). Each consolidation allows TBC to analyze by size and package, because each product is associated with members of the Ounces and Pkg Type attribute dimensions.
- **Market.** Several states make up a region, and four regions make up a market. The states are Connecticut, Florida, Massachusetts, New Hampshire, New York, California, Nevada, Oregon, Utah, Washington, Louisiana, New Mexico, Oklahoma, Texas, Colorado, Illinois, Iowa, Missouri, Ohio, and Wisconsin. Each state consolidates into its respective region—East, West, South, or Central. Each region consolidates into Market.
- **Scenario.** TBC derives and tracks budget data versus actual data. Managers must monitor and track budgets and actuals, as well as the variance and variance percentage between them.
- **Pkg Type.** TBC wants to see the effect that product packaging has on sales and profit. Establishing the Pkg Type attribute dimension enables users to analyze product information based on whether a product is packaged in bottles or cans.
- **Ounces.** TBC sells products in different sizes in ounces in different market areas. Establishing the Ounces attribute dimension helps users to monitor which sizes sell better in which markets.

The following topics present a review of the basics of dimension and member properties and a discussion of how outline design affects performance:

- "Dimension and Member Properties" on page 97
- "Dimension Types" on page 97
- "Member Storage Properties" on page 99
- "Checklist for Dimension and Member Properties" on page 100
- "Designing an Outline to Optimize Performance" on page 100

Dimension and Member Properties

An outline is comprised of dimensions and members. Dimensions and members have specific properties that provide access to built-in functionality. The properties of dimensions and members define the roles of the dimensions and members in the design of the multidimensional structure. These properties include the following:

- Dimension types and attribute associations. See "Dimension Types" on page 97.
- Data storage properties. See "Member Storage Properties" on page 99.
- Consolidation operators. See "Consolidation of Dimensions and Members" on page 104.
- Formulas. See "Formulas and Functions" on page 110.

For a complete list of dimension and member properties, see Chapter 9, "Setting Dimension and Member Properties."

Dimension Types

A dimension type is a property that Analytic Services provides that adds special functionality to a dimension. The most commonly used dimension types are time, accounts, and attribute. This topic uses dimensions of the TBC database to illustrate dimension types.

Figure 39: TBC Dimensions and Related Properties

```
Database:Design
   Year (Type: time)
   Measures (Type: accounts)
   Product
   Market
   Scenario
   Pkg Type (Type: attribute)
   Ounces (Type: attribute)
```

Table 5 defines each Analytic Services dimension type.

Table 5: Dimension Types

Dimension Types	Description
None	Specifies no particular dimension type.
Time	Defines the time periods for which you report and update data. You can tag only one dimension as time. The time dimension enables several accounts dimension functions, such as first and last time balances.
Accounts	Contains items that you want to measure, such as profit and inventory, and makes Analytic Services built-in accounting functionality available. Only one dimension can be defined as accounts. For discussion of two forms of account dimension calculation, see "Accounts Dimension Calculations" on page 108.
Attribute	Contains members that can be used to classify members of another, associated dimension. For example, the Pkg Type attribute dimension contains a member for each type of packaging, such as bottle or can, that applies to members of the Product dimension.
Country	Contains data about where business activities take place. In a country dimension, you can specify the type of currency used in each member. For example, Canada has three markets—Vancouver, Toronto, and Montreal. They use the same currency type, Canadian dollars.
Currency partition	Separates local currency members from the base currency defined in the application. This dimension type is used only in the main database and is only for currency conversion applications. The base currency for analysis may be US dollars, and the local currency members may contain values that are based on the currency type of their region.

Member Storage Properties

With Analytic Services, you can specify data storage properties for members; data storage properties define where and when consolidations are stored. For example, by default, members are tagged as store data. Analytic Services sums the values of store data members and stores the result at the parent level.

You can change the default logic for each member by changing the data storage property tag for the member. For example, you can change a store data member to label only member. Members with the label only tag, for example, do not have data associated with them.

Table 6 describes Analytic Services data storage properties.

Table 6: Analytic Services Data Storage Properties

Storage Properties	Effects on Members
Store data	The member stores data. Store data is the default storage property.
Dynamic Calc	The data associated with the member is not calculated until requested by a user. The calculated data is not stored; it is discarded after the request is completed.
Dynamic Calc and Store	The data associated with the member is not calculated until it is requested by a user. The calculated data is then stored.
Shared member	The data associated with the member comes from another member with the same name.
Never share	The data associated with the member is duplicated with its parent or child if an implied shared relationship exists.
Label only	Although a label only member has no data associated with it, it can still display a value. The label only tag groups members and eases navigation and reporting. Typically, label only members are not calculated. For example, in the Measures dimension, the member Ratios has three children, Margin%, Profit%, and Profit per Ounce. The member Ratios defines a category of members. When consolidated, Margin%, Profit%, and Profit per Ounce do not roll up to a meaningful figure for Ratios. Hence, Ratios is tagged as label only.

Checklist for Dimension and Member Properties

- Can you identify a time dimension?
- Can you identify an accounts dimension?
- Does the data include foreign currencies? If so, did you identify a currency partition dimension?
- Can you identify qualities or characteristics of dimensions that should be defined as separate attribute dimensions?
- What members require special data storage properties?

Designing an Outline to Optimize Performance

When you design an outline, you must position attribute dimensions at the end of the outline. You should position dense dimensions before sparse dimensions.

The position of dimensions in an outline and the storage properties of dimensions can affect two areas of performance—how quickly calculations are run and how long it takes users to retrieve information.

Use the following topics to understand performance optimization basics:

- "Optimizing Query Performance" on page 100
- "Optimizing Calculation Performance" on page 101
- "Meeting the Needs of Both Calculation and Retrieval" on page 102

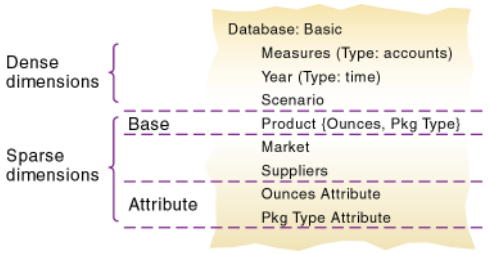
Optimizing Query Performance

To optimize query performance, use the following guidelines when you design an outline:

- If the outline contains attribute dimensions, make sure that the attribute dimensions are the only sparse Dynamic Calc dimensions in the outline.
- In the outline, place the most queried sparse dimensions before the less queried sparse dimensions.

The outline shown in Figure 40 is designed for optimum query performance:

Figure 40: Designing an Outline for Optimized Query Times



- Because the outline contains attribute dimensions, the storage property for standard dimensions and all standard dimensions members is set as store data.
- As the most-queried sparse dimension, the Product dimension is the first of the sparse dimensions. Base dimensions are typically queried more than other dimensions.

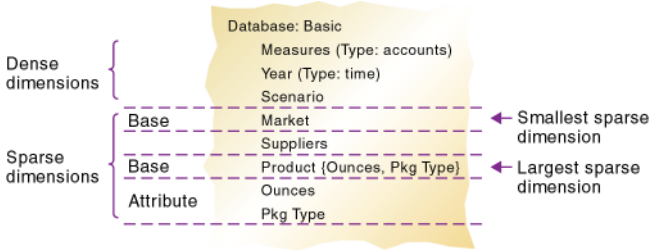
Optimizing Calculation Performance

To optimize calculation performance, order the sparse dimensions in the outline by their number of members, starting with the dimension that contains the fewest members.

For information about factors that affect calculation performance, see "Designing for Calculation Performance" on page 1172.

The outline shown in Figure 41 is designed for optimum calculation performance:

Figure 41: Designing an Outline for Optimized Calculation Times



- The smallest standard dimension that is sparse, Market, is the first of the sparse dimensions in the outline.
- The largest standard dimension that is sparse, Product, is immediately above the first attribute dimension. If the outline did not contain attribute dimensions, the Product dimension would be at the end of the outline.

Meeting the Needs of Both Calculation and Retrieval

Even though they contain the same dimensions, the example outlines of Figure 40 and Figure 41 are different. To determine the best outline sequence for a situation, you must prioritize the data retrieval requirements of the users against the time needed to run calculations on the database. How often do you expect to update and recalculate the database? What is the nature of user queries? What is the expected volume of user queries?

A possible workaround is initially to position the dimensions in the outline to optimize calculation. After you run the calculations, you can manually resequence the dimensions to optimize retrieval. When you save the outline after you reposition its dimensions, choose to restructure the database by index only. Before you run calculations again, remember to resequence the dimensions in the outline to optimize calculation.

Checking System Requirements

After you determine the approximate number of dimensions and members in your Analytic Services database, you are ready to determine the system requirements for the database.

- Make sure that you have enough disk space. See “Determining Disk Space Requirements” on page 1357.
- Make sure that you have enough memory. See “Estimating Memory Requirements” on page 1375.
- Make sure that your caches are set correctly. See Chapter 51, “Optimizing Analytic Services Caches.”

Loading Test Data

Before you can test calculations, consolidations, and reports, you need data in the database. During the design process, loading mocked-up data or a subset of real data provides flexibility and shortens the time required to test and analyze results.

Detailed instructions for loading data are in the following chapters:

- Chapter 16, “Understanding Data Loading and Dimension Building”
- Chapter 17, “Creating Rules Files”
- Chapter 18, “Using a Rules File to Perform Operations on Records, Fields, and Data”
- Chapter 19, “Performing and Debugging Data Loads or Dimension Builds”
- Chapter 20, “Understanding Advanced Dimension Building Concepts”

After you run your preliminary test, if you are satisfied with your database design, test the loading of the complete set of real data with which you will populate the final database, using the test rules files if possible. This final test may reveal problems with the source data that you did not anticipate during earlier phases of the database design process.

Defining Calculations

Calculations are essential to derive certain types of data. Data that is derived from a calculation is called *calculated data*; basic noncalculated data is called *input data*.

The following topics use the Product and Measures dimensions of the TBC application to illustrate several types of common calculations that are found in many Analytic Services databases:

- “Consolidation of Dimensions and Members” on page 104
- “Tags and Operators on Example Measures Dimension” on page 107
- “Accounts Dimension Calculations” on page 108
- “Formulas and Functions” on page 110

- “Dynamic Calculations” on page 112
- “Two-Pass Calculations” on page 113

For details on Analytic Services calculations, see the following chapters:

- Chapter 21, “Calculating Analytic Services Databases”
- Chapter 30, “Developing Custom-Defined Calculation Functions”
- Chapter 55, “Optimizing with Intelligent Calculation”

Consolidation of Dimensions and Members

When you define members of standard dimensions, Analytic Services automatically tags the members with the addition (+) consolidator, meaning that during consolidation members are added. As appropriate, you can change a member consolidation property to one of the following operators: -, *, /, %, and ~ (no consolidation).

Consolidation is the most frequently used calculation in Analytic Services. This topic uses the Product dimension to illustrate consolidations.

The TBC application has several consolidation paths:

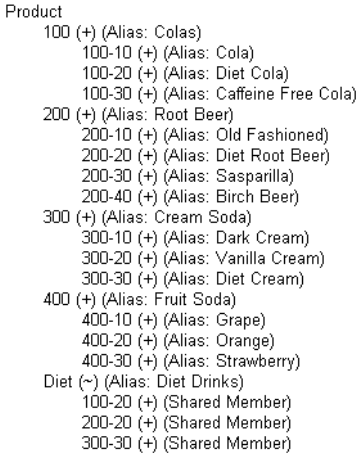
- Individual products roll up to product families, and product families consolidate into Product. The TBC outline also requires multiple consolidation paths; some products must consolidate in multiple categories.
- States roll up to regions, and regions consolidate into Market.
- Months roll up into quarters, and quarters consolidate into Year.

The following topics discuss consolidation in greater detail:

- “Effect of Position and Operator on Consolidation” on page 105
- “Consolidation of Shared Members” on page 106
- “Checklist for Consolidation” on page 107

Consolidation operators define how Analytic Services rolls up data for each member in a branch to the parent. For example, using the default operator (+), Analytic Services adds 100-10, 100-20, and 100-30 and stores the result in their parent, 100, as shown in Figure 42.

Figure 42: TBC Product Dimension



The Product dimension contains mostly (+), operators, which indicate that each group of members is added and rolled up to the parent. Diet has a tilde (~), which indicates that Analytic Services does not include the Diet member in the consolidation to the parent, Product. The Diet member consists entirely of members that are shared or duplicated. The TBC product management group wants to be able to isolate Diet drinks in reports, so TBC created a separate Diet member that does not impact overall consolidation.

The following topics discuss consolidation in more detail:

- “Effect of Position and Operator on Consolidation” on page 105
- “Consolidation of Shared Members” on page 106
- “Checklist for Consolidation” on page 107

Effect of Position and Operator on Consolidation

Analytic Services calculates the data of a branch in top-down order. For example, if you have, in order, two members tagged with an addition symbol (+) and a third member tagged with a multiplication symbol (*). Analytic Services adds the first two and multiplies the sum by the third.

Be aware that Analytic Services always begins with the top member when it consolidates, so the order and the labels of the members is very important. For an example of how Analytic Services applies operators, see "Calculating Members with Different Operators" on page 161.

Table 7 shows the Analytic Services consolidation operators.

Table 7: Consolidation Operations

Operator	Description
+	The default operator. When a member has the + operator, Analytic Services adds the member to the result of previous calculations performed on members of the branch.
–	When a member has the – operator, Analytic Services multiplies the member by -1 and then adds the product to the result of previous calculations performed on members of the branch.
*	When a member has the * operator, Analytic Services multiplies the member by the result of previous calculations performed on members of the branch.
/	When a member has the / operator, Analytic Services divides the member into the result of previous calculations performed on members of the branch.
%	When a member has the % operator, Analytic Services divides the member into the sum of previous calculations performed on members of the branch. The result is multiplied by 100.
~	When a member has the ~ operator, Analytic Services does not use it in the consolidation to its parent.

Consolidation of Shared Members

Shared members also affect consolidation paths. The shared member concept enables two members with the same name to share the same data. The shared member stores a pointer to data contained in the other member, so Analytic Services stores the data only once. Shared members must be in the same dimension. Data can be shared by two or more members.

Checklist for Consolidation

Use the following checklist to help define consolidation:

- Have you identified the consolidations in the outline?
- Did you tag each member with the proper consolidation operator?
- Did you specify a shared member tag for designated members?
- Would shared members be more efficient if designed within an attribute dimension (other than shared)?

Tags and Operators on Example Measures Dimension

The Measures dimension is the most complex dimension in the TBC outline because it uses both time and accounts data. It also contains formulas and special tags to help Analytic Services calculate the outline. This topic discusses the formulas and tags that TBC included in the Measures dimension (the dimension tagged as accounts).

Take a moment to look closely at the Measures dimension tags defined by TBC (in Figure 43). Many of the properties of the Measures dimension are discussed in previous topics of this chapter: positive (+), negative (–), and tilde (~) consolidation operators as well as accounts and label only tags:

Figure 43: TBC Measures Dimension

```
Measures Accounts (Label Only)
    Profit (+) (Dynamic Calc)
        Margin (+) (Dynamic Calc)
            Sales (+)
            COGS (-) (Expense Reporting)
        Total Expenses (-) (Dynamic Calc) (Expense Reporting)
            Marketing (+) (Expense Reporting)
            Payroll (+) (Expense Reporting)
            Misc (+) (Expense Reporting)
    Inventory (~) (Label Only)
        Opening Inventory (+) (TB First) (Expense Reporting)
        Additions (~) (Expense Reporting)
        Ending Inventory (~) (TB Last) (Expense Reporting)
    Ratios (~) (Label Only)
        Margin % (+) (Dynamic Calc) (Two Pass Calc) Margin % Sales;
        Profit % (~) (Dynamic Calc) (Two Pass Calc) Profit % Sales;
        Profit per Ounce (~) Profit/@ATTRIBUTEVAL(Ounces);
```

- The Inventory and Ratios member names assist the user in data navigation and do not contain data and, therefore, receive a label only tag.
- The Measures dimension itself has a label only tag. Some members of Measures have a Dynamic Calc tag. Dynamic calculations are discussed in "Dynamic Calculations" on page 112.
- Some members of Measures have a time balance tag (TB First or TB Last). Time balance tags are discussed in "Setting Time Balance Properties" on page 155.

Accounts Dimension Calculations

This topic discusses two forms of calculations for a dimension tagged as accounts:

Time Balance Properties

Note the two tags in the Measures dimension of Table 9—TB first and TB last. These tags, called time balance tags or properties, provide instructions to Analytic Services about how to calculate the data in a dimension tagged as accounts. To use the tags, you must have a dimension tagged as accounts and a dimension tagged as time. The first, last, average, and expense tags are available exclusively for use with accounts dimension members.

In the TBC Measures dimension, Opening Inventory data represents the inventory that TBC carries at the beginning of each month. The quarterly value for Opening Inventory is equal to the Opening value for the quarter. Opening Inventory requires the time balance tag, TB first.

Ending Inventory data represents the inventory that TBC carries at the end of each month. The quarterly value for Ending Inventory is equal to the ending value for the quarter. Ending Inventory requires the time balance tag, TB last. Table 8 shows the time balance tags for the accounts dimension.

Table 8: Accounts Member Tags

Tags	Description
Time Balance Last	The value for the last child member is carried to the parent. For example, March is carried to Qtr1.
Time Balance First	The value for the first child is carried to the parent. For example, Jan is carried to Qtr1.

Table 9 shows how consolidation in the time dimension is affected by time balance properties in the accounts dimension; details are shown only for first quarter:

Table 9: TBC Consolidations Affected by Time Balance Properties

Accounts -> Time	Jan	Feb	Mar	Qtr1	Year
Accounts Member1	11	12	13	36	Qtr1 + Qtr2 + Qtr3 + Qtr4
Accounts Member2 (TB First)	20	25	21	20	20
Accounts Member3 (TB Last)	25	21	30	30	Value of Qtr4

Normally, the calculation of a parent in the time dimension is based on the consolidation and formulas of children of the parent. However, if a member in an accounts branch is marked as TB First, then any parent in the time dimension matches the member marked as TB First.

For examples of the use of time balance tags, see “Setting Time Balance Properties” on page 155.

Variance Reporting

One of the TBC Analytic Services requirements is the ability to perform variance reporting on actual versus budget data. The variance reporting calculation requires that any item that represents an expense to the company must have an expense reporting tag. Inventory members, Total Expense members, and the COGS member each receive an expense reporting tag for variance reporting.

Analytic Services provides two variance reporting properties—expense and non-expense. The default is non-expense. Variance reporting properties define how Analytic Services calculates the difference between actual and budget data in members with the @VAR or @VARPER function in their member formulas.

When you tag a member as expense, the @VAR function calculates Budget - Actual. For example, if the budgeted amount is $100 and the actual amount is $110, the variance is -10.

Without the expense reporting tag, the @VAR function calculates Actual - Budget. For example, if the budgeted amount is $100 and the actual amount is $110, the variance is 10.

Formulas and Functions

You can define formulas to calculate relationships between members in the database outline. You can either apply the formulas to members in the outline, or you can place the formulas in a calculation script. This topic explains how TBC optimized the performance of its database by using formulas.

Functions are predefined routines that perform specialized calculations and return sets of members or sets of data values. Formulas are composed of operators and functions, as well as dimension names, member names, and numeric constants.

Analytic Services supports the following operators:

- Mathematical operators that perform arithmetic operations
- Conditional operators that build logical conditions into calculations
- Cross-dimensional operators that point to data values of specific database member combinations

The Analytic Services functions include over 100 predefined routines to extend the calculation capabilities of Analytic Services. Analytic Services supports the following functions:

- Boolean functions, which provide a conditional test by returning a TRUE or FALSE value
- Mathematical functions, which perform specialized mathematical calculations
- Relationship functions, which look up data values within a database during a calculation based on the position of the current member.
- Range functions, which declare a range of members as an argument to another function or to a command
- Financial functions, which perform specialized financial calculations
- Member set functions, which are based on a specified member and which generate lists of members
- Allocation functions, which allocate values that are input at a parent level across child members
- Forecasting functions, which manipulate data for the purpose of smoothing data, interpolating data, or calculating future values
- Statistical functions, which calculate advanced statistics
- Date and time functions, which use date and time characteristics in calculation formulas
- Calculation mode functions, which specify the calculation mode that Analytic Services uses to calculate a formula

The Measures dimension uses the following formulas:

- Margin = Sales - COGS
- Total Expenses = Marketing + Payroll + Miscellaneous
- Profit = Margin - Total Expenses
- Profit % = Profit % Sales
- Margin % = Margin % Sales
- Profit per Ounce = Profit / @ATTRIBUTEVAL(@NAME(Ounces))

Analytic Services uses consolidation operators to calculate the Margin, Total Expenses, and Profit members. The Margin% formula uses a % operator, which means “express Margin as a percentage of Sales.” The Profit% formula uses the same % operator. The Profit per Ounce formula uses a division operator (/) and a function (@ATTRIBUTEVAL) to calculate profitability by ounce for products sized in ounces.

Note: In the Profit per Ounce formula, the @NAME function is also used to process the string “Ounces” for the @ATTRIBUTEVAL function.

For a complete list of operators, functions, and syntax, see the *Technical Reference*. For a comprehensive discussion of how to use formulas, see Chapter 22, “Developing Formulas”.

Dynamic Calculations

When you design the overall database calculation, you may want to define a member as a Dynamic Calc member. When you tag a member as Dynamic Calc, Analytic Services calculates the combinations of that member when you retrieve the data, instead of pre-calculating the member combinations during the regular database calculation. Dynamic calculations shorten regular database calculation time but may increase retrieval time for dynamically calculated data values.

As shown in Figure 44, the TBC Measures dimension contains several members that are tagged as Dynamic Calc—Profit, Margin, Total Expenses, Margin %, and Profit %.

Figure 44: TBC Measures Dimension, Dynamic Calc Tags

```
Measures Accounts (Label Only)
    Profit (+) (Dynamic Calc)
        Margin (+) (Dynamic Calc)
            Sales (+)
            COGS (-) (Expense Reporting)
        Total Expenses (-) (Dynamic Calc) (Expense Reporting)
            Marketing (+) (Expense Reporting)
            Payroll (+) (Expense Reporting)
            Misc (+) (Expense Reporting)
    Inventory (~) (Label Only)
        Opening Inventory (+) (TB First) (Expense Reporting)
        Additions (~) (Expense Reporting)
        Ending Inventory (~) (TB Last) (Expense Reporting)
    Ratios (~) (Label Only)
        Margin % (+) (Dynamic Calc) (Two Pass Calc) Margin % Sales;
        Profit % (~) (Dynamic Calc) (Two Pass Calc) Profit % Sales;
        Profit per Ounce (~) Profit/@ATTRIBUTEVAL(Ounces);
```

When an overall database calculation is performed, the Dynamic Calc members and their corresponding formulas are not calculated. Rather, the members are calculated when a user requests them, for example, from Spreadsheet Add-in. Analytic Services does not store the calculated values; it recalculates the values for any subsequent retrieval. However, you can choose to store dynamically calculated values after the first retrieval.

To decide when to calculate data values dynamically, consider your priorities in the following areas:

- Optimum regular calculation time (batch calculation)
- Low disk space usage
- Reduced database restructure time
- Speedy data retrieval for users
- Reduced backup time

For a comprehensive discussion of dynamic calculation, see Chapter 25, "Dynamically Calculating Data Values".

Two-Pass Calculations

In the TBC database, both Margin % and Profit % contain the label two-pass. This default label indicates that some member formulas need to be calculated twice to produce the desired value. The two-pass property works only on members of the dimension tagged as accounts and on members tagged as Dynamic Calc and Dynamic Calc and Store. The following examples illustrate why Profit % (based on the formula Profit%Sales) has a two-pass tag.

Analytic Services loads data into the system as follows:

Measures -> Year	Jan	Feb	Mar	Qtr1
Profit	100	100	100	
Sales	1000	1000	1000	
Profit %				

Analytic Services calculates Measures first. The data then looks as follows:

Measures -> Year	Jan	Feb	Mar	Qtr1
Profit	100	100	100	
Sales	1000	1000	1000	
Profit %	10%	10%	10%	

Next, Analytic Services calculates the Year dimension. The data rolls up across the dimension.

Measures -> Year	Jan	Feb	Mar	Qtr1
Profit	100	100	100	300
Sales	1000	1000	1000	3000
Profit %	10%	10%	10%	30%

The result in Profit % -> Qtr1 of 30% is not correct. However, because TBC tagged Profit% as two-pass calculation, Analytic Services recalculates profit percent at each occurrence of the member Profit %. The data is then correct and is displayed as follows:

Measures -> Year	Jan	Feb	Mar	Qtr1
Profit	100	100	100	300
Sales	1000	1000	1000	3000
Profit %	10%	10%	10%	10%

Checklist for Calculations

Use the following checklist when you define a calculation:

- Does the default calculation logic achieve accurate results?
- Which members require formulas?
- Which members require time balance tags?

- Which members require variance reporting?
- Which members require two-pass calculation?
- Which members can be tagged as Dynamic Calc?

Note: The triggers feature provided by Analytic Services enables efficient monitoring of data changes in a database. For more information, see "Understanding Triggers Definitions" on page 130. Triggers is licensed separately from Analytic Services.

Defining Reports

To be sure the design meets user information requirements, you need to view data as users view it. Users typically view data through spreadsheets, printed reports, or reports published on the Web. There are many tools available through Hyperion and Hyperion partners for producing the reporting systems that users use.

Analytic Services provides several tools that can help you during the design process to display and format data quickly and to test whether the database design meets user needs. You can use Administration Services Console Report Script Editor to write report scripts quickly. Those familiar with spreadsheets can use the Spreadsheet Add-in or Spreadsheet Services (Spreadsheet Services requires Deployment Services).

During the design phase, check for the following things:

- Grouping and sequencing of data. Do the intersections enabled by the design provide the data that users need?
- Levels of totals. What consolidation levels are required by, for example, a Spreadsheet Add-in user who drills down and up through the hierarchy of the outline design?
- Attribute reporting. Does the database design facilitate an analysis that is based on the characteristics or attributes of specific dimensions or members? For example, do you need to compare sales by specific combinations of size and packaging, such as comparing the sales of 16-ounce bottled colas with the sales of 32-ounce bottled colas?

If you provide predesigned reports for users, now is the time to use the appropriate tool to create those reports against the test data. The reports that you design should provide information that meets your original objectives. The reports should be easy to use. They should provide the right combinations of data and the right

amount of data. Reports with too many columns and rows are very hard to use. It may be better to create a number of different reports instead of one or two all-inclusive reports.

Verifying the Design

After you analyze the data and create a preliminary design, you need to check all aspects of the design with the users. You should have already checked to see if the database satisfies the users' analysis and reporting needs. Make sure that you check with the users to ensure that the database satisfies all of their goals.

Do the calculations give them the information they need? Are they able to generate reports quickly? Are they satisfied with consolidation times? In short, ask users if the database works for them.

Near the end of the design cycle, you need to test with real data. Does the outline build correctly? Does all data load? If the database fails in any area, repeat the steps of the design cycle to identify the cause of the problem.

Analytic Services provides several sources of information to help you isolate problems. Sources include application and Analytic Server logs, exception logs, and database information accessible from Administration Services. Look at documentation topics relevant to your problem; for example, topics about security, calculations, reports, or general error messages. You can also use the index of this guide to find help for solving problems. Look up such terms as troubleshooting, logs, optimizing, performance, recovery, resources, errors, and warnings.

Most likely, you will need to repeat one or more steps of the design process to arrive at the ideal database solution.

Chapter 6

About Essbase Administration Services

Essbase Administration Services is the cross-platform administration tool for Essbase Analytic Services, replacing Application Manager. Administration Services consists of a Java middle-tier server, called Administration Server, and a Java client console, called Administration Services Console.

Administration Services Console is the graphical user interface that enables administrators to manage the Analytic Services environment from a single navigation tree, called Enterprise View. The console provides wizards, editors, and other tools to help administrators view, manage, and maintain a unique set of Essbase Analytic Servers. The console includes a data preview grid that enables you to preview data without having to switch from the console to another program.

Essbase Administration Services provides its own set of documentation, including an installation guide, a context-sensitive online help system, a developer's guide, a Java API Reference, and a readme. To view documentation, first start Administration Server and then launch the Essbase Administration Services Information Map (`eas\doc_launcher.htm` under the Essbase Administration Services installation directory).

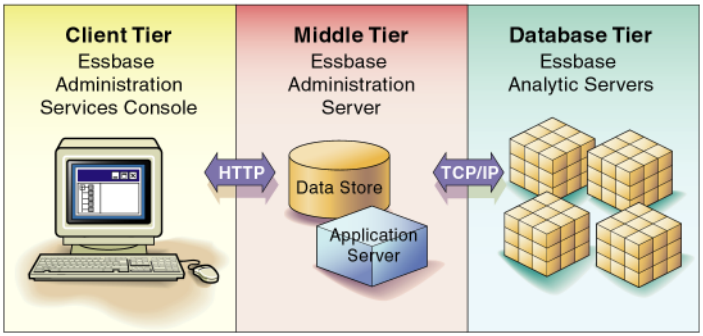
For information about release compatibility, platform support and system requirements, and installation instructions, see *Essbase Administration Services Installation Guide*. For procedural instructions for using Administration Services to manage Essbase, see *Essbase Administration Services Online Help*.

Administration Services Architecture

Administration Services works with Analytic Servers in a three-tiered system that consists of a client user interface, a middle-tier server, and one or more Analytic Servers. The middle tier coordinates interactions and resources between the user

interface and Analytic Servers. The three tiers may or may not be on the same computer or platform. The three tiers include the following components, as illustrated below:

Figure 45: Administration Services Architecture



- Client tier (Administration Services Console): A Java-based client console provides a user interface to manage the Analytic Services environment.
- Middle tier (Administration Server): A Java-based server maintains communication, session, and security information for connections to Analytic Servers.
- Database tier (Analytic Server): One or more Analytic Servers store and process multidimensional database information. Analytic Servers are installed separately from Administration Services.

Note: For information about which platforms are supported for the client and middle tier components, see *Essbase Administration Services Installation Guide*.

Deploying Administration Services

Administration Services can be deployed in a variety of scenarios. For example, you can install Analytic Server on a computer running UNIX and install Administration Server and Administration Services Console on a computer running Windows. You can also install Administration Server and Administration Services Console on separate computers and platforms. The middle tier Administration Server also supports the substitution of certain third-party products within the existing framework (for example, application servers and relational databases).

For complete information about deployment scenarios and guidelines, see *Essbase Administration Services Installation Guide*.

Starting Administration Services

To start Administration Services, first start Administration Server, and then start Administration Services Console. For instructions, see "Starting Administration Services" in *Essbase Administration Services Installation Guide*.

If you have enabled an Analytic Server for remote start, you can start that Analytic Server remotely from Enterprise View in Administration Services Console. To enable this functionality, you need to configure and start the Remote Start Service on the Analytic Server machine. For instructions, see "Starting Analytic Server Remotely from Administration Services Console" on page 920.

About Administration Services Users

Existing Analytic Services users cannot use Administration Services until they have also been created as users on Administration Server. You can use the User Setup Wizard to step you through the process of creating Administration Server users and associating them with the appropriate Analytic Servers. You do not to create spreadsheet users on the Administration Server.

➤ To create Administration Services users, see "User Setup Wizard" in *Essbase Administration Services Online Help*.

Connecting to Administration Services

In Administration Services, connections to individual Analytic Servers are handled by the middle tier Administration Server. When you start Administration Services, you are automatically connected to each Analytic Server you have added to Enterprise View (if Analytic Server is started). For information about how Analytic Server connections are established, see "About Analytic Services Connections and Ports" in *Essbase Administration Services Online Help*.

You can connect to different releases of Analytic Server simultaneously from Administration Services Console.

Your Administration Services username and password may be different than your Analytic Server username and password. If you do not know your Administration Services username and password, see your Administration Services administrator for more information.

After your initial connection to Administration Services, you can use the User Setup Wizard to create Administration Services users and add Analytic Servers to each user's Enterprise View. For more information, see "Connecting to Administration Services" in *Essbase Administration Services Online Help*.

Adding Administration Servers to Enterprise View

Each time you connect to Administration Services, the Administration Servers you have chosen are displayed in Enterprise View, which is the navigation tree in the left navigation panel. Each user can populate Enterprise View with a unique set of Administration Servers. You can use the following methods to add Analytic Servers to Enterprise View:

➤ To add Administration Servers to Enterprise View, see "Adding Administration Servers to Enterprise View" in *Essbase Administration Services Online Help*.

Adding Analytic Servers to Enterprise View

Each time you connect to Administration Services, the Analytic Servers you have chosen are displayed in Enterprise View, which is the navigation tree in the left navigation panel. Each user can populate Enterprise View with a unique set of Analytic Servers. You can use the following methods to add Analytic Servers to Enterprise View:

- In Enterprise View, right-click the Essbase Analytic Servers node, and select Add Analytic Server.
- Select Wizards > User Setup and follow the steps in the wizard.
- For an existing user, edit the user's properties.

You can also create custom views of the Enterprise View tree in separate tabs in the navigation panel. For more information, see "About Custom Views" in *Essbase Administration Services Online Help*.

➤ To add Analytic Servers to Enterprise View, see “Adding Analytic Servers to Enterprise View” in *Essbase Administration Services Online Help*.

About Analytic Server Connections and Ports

The number of ports available for an Analytic Server represents the number of licensed concurrent connections. Analytic Services provides one reserve port for the system administrator. A system administrator uses the reserve port to log out one or more users when all other ports are in use. For more information about Analytic Services ports, see Chapter 41, “Running Analytic Servers, Applications, and Databases.”

In Administration Services, a port is in use only when an Analytic Server connection is established. For information about how connections (ports) are established and released, see “About Analytic Services Connections and Ports” in *Essbase Administration Services Online Help*.

About Administration Server

The middle tier Administration Server provides business logic to support cross-server operations, persistence of user preferences, and access to Analytic Servers. A system administrator creates users on Administration Server, and then Administration Server manages their connections to Analytic Services.

In Enterprise View, the node name for Administration Server is the same as the server computer name.

Administration Server has several configurable communication ports. These ports are different from Analytic Server ports. If one of the default communication ports is in use by another application, you need to specify another port value in order to run Administration Server.

Note: If you change the value for the Administration Server port, you must specify the new port value when you log in to the Administration Services Console.

➤ To change a default port value, see “Specifying Communication Ports for Administration Server” in *Essbase Administration Services Online Help*.

Part

II

Designing and Creating Applications and Databases

Part II describes how to design and create basic Essbase Analytic Services applications and databases. This part includes information on dimension and member properties, linked reporting objects, partitioning, currency conversion, and hybrid analysis. Part II contains the following chapters:

- Chapter 7, "Creating Applications and Databases," describes how to create and manage applications and databases and application and database-level objects, such as substitution variables and location aliases.
- Chapter 8, "Creating and Changing Database Outlines," describes how to create and modify database outlines, including adding and positioning dimensions and members and verifying and saving outlines.
- Chapter 9, "Setting Dimension and Member Properties," illustrates how to set properties for the dimensions and members in an outline.
- Chapter 10, "Working with Attributes," describes attribute dimensions and members, how to define them, and how they are calculated.
- Chapter 11, "Linking Objects to Analytic Services Data," describes how to link various kinds of data with any cell in a Analytic Services database.
- Chapter 12, "Designing and Building Currency Conversion Applications," describes how to create and calculate currency conversion applications.
- Chapter 13, "Designing Partitioned Applications," explains the advantages, disadvantages, and requirements for each partition type.
- Chapter 14, "Creating and Maintaining Partitions," describes how to create and maintain partitions.
- Chapter 15, "Accessing Relational Data with Hybrid Analysis," describes how to integrate a relational database with an Analytic Services database using the Hybrid Analysis feature.

Chapter 7

Creating Applications and Databases

An Analytic Services application is a container for a database and its related files. This chapter provides an overview of Analytic Services applications and databases and explains how to create applications and databases and some Analytic Services objects, including substitution variables and location aliases. For information on everyday management of applications, databases, and their associated files, see the optimization and system administration information in this guide.

Note: The information in this chapter is designed for block storage databases. Some of the information is not relevant to aggregate storage databases. For detailed information on the differences between aggregate and block storage, see Chapter 57, "Comparison of Aggregate and Block Storage." For information on creating aggregate storage applications, see Chapter 58, "Aggregate Storage Applications, Databases, and Outlines."

This chapter includes the following topics:

Process for Creating Applications and Databases

➤ To create an application and database, follow these steps:

1. Design the application. See "Quick Start for Implementing Analytic Services" on page 53,
2. Create a new application. See "Creating a New Application" on page 131.
3. Create a new database. See "Creating a New Database" on page 132.
4. If necessary, set substitution variables at the Analytic Server, application, or database level. See "Using Substitution Variables" on page 133.
5. If necessary, set a location alias for the database. See "Using Location Aliases" on page 136.
6. Create the outline. See Chapter 8, "Creating and Changing Database Outlines."

For more information about applications and database, see "Understanding Applications and Databases" on page 126 and "Understanding Database Objects" on page 127.

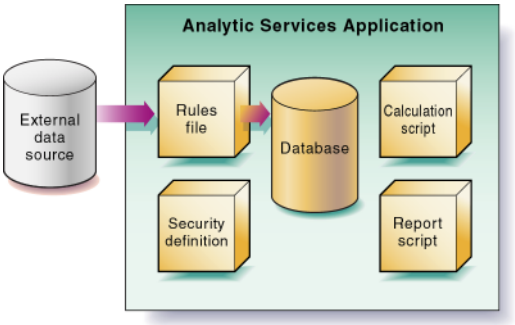
Understanding Applications and Databases

An Analytic Services application is a management structure that contains one or more Analytic Services databases and related files. Analytic Services applications and databases reside on an Analytic Server. The server machine can store multiple applications.

An Analytic Services database is a data repository that contains a multidimensional data storage array. A multidimensional database supports multiple views of data so that users can analyze the data and make meaningful business decisions. For more information about multidimensional databases, see "Understanding Multidimensional Databases" on page 31. For more information about how Analytic Services stores data, see "Storage Allocation" on page 1037.

This diagram shows the relationships among the parts of an application:

Figure 46: Parts of an Analytic Services Application



Understanding Database Objects

Files that are related to databases are called *objects*. Database objects perform actions against one or more Analytic Services databases, such as defining calculations or reporting against data. By default, objects are stored in their associated database folder on the Analytic Server. They can also be saved to a client machine or to other available network directories. However, you cannot load data or calculate data on a client machine.

In Analytic Services, the common types of objects include the following:

- A database outline (a storage structure definition)
- Data sources
- Rules for loading data and building dimensions dynamically (rules files)
- Scripts that define how to calculate data (calculation scripts)
- Scripts that generate reports on data (report scripts)
- Security definitions
- Linked reporting objects
- Partition definitions

Some of these objects are optional, such as calculation scripts and linked reporting objects. For a complete list of application and database file types, see “Application and Database File Types” on page 953.

In Administration Services Console, database objects are displayed under their associated applications or database in the Enterprise View tree.

Understanding Database Outlines

Database outlines define the structure of a multidimensional database, including all the dimensions, members, aliases, properties, types, consolidations, and mathematical relationships. The structure defined in the outline determines how data is stored in the database.

When a database is created, Analytic Services creates an outline for that database automatically. The outline has the same name as the database (*dbname*.otl). For example, when the Basic database is created within the Sample application, an outline is created in the following directory:

ARBORPATH/app/sample/basic/basic.otl

For information about creating outlines, see "Creating a New Database" on page 132 and Chapter 8, "Creating and Changing Database Outlines."

Understanding Data Sources

A data source is external data that is loaded into an Analytic Services database. The common types of data sources include the following:

- Text files
- Spreadsheet files
- Spreadsheet audit log files
- External databases, such as an SQL database

For a list of supported data sources, see "Supported Data Sources" on page 357.

Understanding Rules Files for Data Load and Dimension Build

An Analytic Services database contains no data when it is first created. *Data load rules files* are sets of operations that Analytic Services performs on data from an external data source file as the data is loaded, or copied, into the Analytic Services database. *Dimension build rules files* create or modify the dimensions and members in an outline dynamically based on data in an external data source. Rules

files are typically associated with a particular database, but you can define rules for use with multiple databases. A single rules file can be used for both data loads and dimension builds. Rules files have the `.RUL` extension.

For information about creating rules files, see "Rules Files" on page 364 and Chapter 17, "Creating Rules Files."

Understanding Calculation Scripts

Calculation scripts are text files that contain sets of instructions telling Analytic Services how to calculate data in the database. Calculation scripts perform different calculations than the consolidations and mathematical operations that are defined in the database outline. Because calculation scripts perform specific mathematical operations on members, they are typically associated with a particular database. You can, however, define a calculation script for use with multiple databases. Calculation scripts files have the `.CSC` extension.

For information about creating calculation scripts, see Chapter 27, "Developing Calculation Scripts."

Understanding Report Scripts

Report scripts are text files that contain data retrieval, formatting, and output instructions to create a report from the database. Report scripts are typically associated with a particular database, but you can define a report script for use with multiple databases. Report scripts have the `.REP` extension.

For information about creating report scripts, see Chapter 32, "Developing Report Scripts."

Understanding Security Definitions

Analytic Services provides a comprehensive system for managing access to applications, databases, and other objects. Each application and database contains its own security definitions that restrict user access.

For information about setting up and maintaining security information, see Chapter 36, "Managing Security for Users and Applications."

Understanding Linked Reporting Objects

A linked reporting object is an object associated with a specific data cell in an Analytic Services database. Linked reporting objects can enhance data analysis capabilities by providing additional information on a data point.

A linked reporting object can be any of the following:

- A paragraph of descriptive text (a “cell note”)
- A separate file that contains text, audio, video, or graphics
- A Uniform Resource Locator (URL) for a Web site
- A link to data in another Analytic Services database

For a comprehensive discussion about using linked reporting objects, see Chapter 11, “Linking Objects to Analytic Services Data.”

Understanding Spreadsheet Queries

Within Spreadsheet Add-in, users can create and save queries using Query Designer (EQD). The queries can be accessed at a later time by any user with access to the query. Query files created using Query Designer have the `.EQD` extension.

For more information, see the *Essbase Spreadsheet Add-in User’s Guide* for Excel.

Understanding Member Select Definitions

Within Spreadsheet Add-in, users can define and save member retrievals with the member select feature. Member specification files have the `.SEL` extension.

For more information, see the *Essbase Spreadsheet Add-in User’s Guide* for Excel.

Understanding Triggers Definitions

The *triggers* feature provided by Analytic Services enables efficient monitoring of data changes in a database.Triggers is licensed separately from Analytic Services. If data breaks rules that you specify in a trigger, Analytic Services can log relevant information in a file or, for some triggers, can send an email alert (to a user or

system administrator). For example, you might want to notify the sales manager if, in the Western region, sales for a month fall below sales for the equivalent month in the previous year.

For information on designing, creating, and administering triggers, see "Monitoring Data Changes Using Triggers" on page 971.

Creating Applications and Databases

Since applications contain one or more databases, first create an application and then create databases. If desired, annotate the databases. The following sections describe how to create applications, databases, and database notes:

Creating a New Application

When you create an application on the Analytic Server, Analytic Services creates a subdirectory for the application on the Analytic Server in the *ARBORPATH*/app directory. The new subdirectory has the same name as the application; for example, essbase/app/app1. In Administration Services Console, applications and databases are displayed in a tree structure in Enterprise View.

Be sure to consult "Rules for Naming Applications and Databases" on page 133 before entering the application name.

You can also create a new application that is a copy of an existing application. For more information, see "Copying or Migrating Applications" on page 958.

➤ To create a new application, use any of the following methods:

Tool	Topic	Location
Administration Services	Creating Applications	*Essbase Administration Services Online Help*

Tool	Topic	Location
MaxL	**create application**	*Technical Reference*
ESSCMD	CREATEAPP	*Technical Reference*

Creating a New Database

When you create a database, Analytic Services creates a subdirectory for the database within the application directory. The new subdirectory has the same name as the database; for example, `essbase/app/app1/db1`. In Administration Services Console, applications and databases are displayed in a tree structure in Enterprise View.

You can create normal databases or currency databases. For more information on currency databases, see Chapter 12, "Designing and Building Currency Conversion Applications."

Be sure to consult "Rules for Naming Applications and Databases" on page 133 before entering the database name.

➤ To create a new database, use any of the following methods:

Tool	Topic	Location
Administration Services	Creating Databases	*Essbase Administration Services Online Help*
MaxL	**create database**	*Technical Reference*
ESSCMD	CREATEDB	*Technical Reference*

Annotating a Database

A database note can provide useful information in situations where you need to broadcast messages to users about the status of a database, deadlines for updates, and so on. Users can view database notes in Spreadsheet Add-in. In Excel, for example, users use the Note button in the Connect dialog box.

➤ To annotate a database, see "Annotating Databases" in the *Essbase Administration Services Online Help*.

Rules for Naming Applications and Databases

When naming applications and databases, follow these rules:

- Use no more than 8 bytes when naming non-Unicode-mode applications and databases; use no more than 30 characters when naming Unicode-mode applications and databases.
- Do not use spaces anywhere in the name.
- Do not use the following special characters anywhere in the name:

*	asterisks	+	plus signs
\	backslashes	?	question marks
[]	brackets	"	double quotation marks
:	colons	;	semicolons
,	commas	‘	single quotation marks
=	equal signs	/	forward slashes
>	greater than signs	tabs	
<	less than signs	|	vertical bars
.	periods		

Enter the name in the case you want it to appear in. The application or database name will be created exactly as you enter it. If you enter the name as all capital letters (for instance, NEWAPP), Analytic Services will not automatically convert it to upper and lower case (for instance, Newapp).

Using Substitution Variables

Substitution variables act as global placeholders for information that changes regularly; each variable has a value assigned to it. The value can be changed at any time by the database designer; thus, manual changes are reduced.

For example, many reports depend on reporting periods; if you generate a report based on the current month, you have to update the report script manually every month. With a substitution variable, such as `CurMnth`, set on the server, you can change the assigned value each month to the appropriate time period. When you use the variable name in a report script, the information is dynamically updated when you run the final report.

You can use substitution variables in calculation scripts, report scripts, or in Spreadsheet Add-in. You cannot use substitution variables in formulas that you apply to the database outline. For information about using substitution variables, refer to the following chapters:

- For calculation scripts, see Chapter 27, "Developing Calculation Scripts."
- For reports, see Chapter 32, "Developing Report Scripts."
- For Spreadsheet Add-in, see the *Essbase Spreadsheet Add-in User's Guide* for Excel.

You can set substitution variables on the Analytic Server using Administration Services, MaxL, or ESSCMD. Set the variable at any of the following levels:

- Analytic Server—providing access to the variable from all applications and databases on the Analytic Server
- Application—providing access to the variable from all databases within the application
- Database—providing access to the variable within the specified database

Rules for Setting Substitution Variable Names and Values

Keep in mind the following rules when setting substitution variables:

- The substitution variable name must be composed of alphanumeric characters or underscores (_) and cannot exceed the limit specified in Appendix A, "Limits."
- The substitution variable name cannot include non-alphanumeric characters, such as hyphens (-), asterisks (*), and slashes (/).
- The substitution variable value cannot exceed 256 characters. You can use any combination of characters in the value name. The value may contain any character except the leading ampersand (&).
- If the substitution variable value is numeric, you must enclose it in quotes. For example, if the variable name is Month and its corresponding value is 01 (corresponding to January), place quotes around 01 ("01").

Setting Substitution Variables

You can set substitution variables on the Analytic Server at the server, application, or database level. Be sure to consult "Rules for Setting Substitution Variable Names and Values" on page 134 before setting a substitution variable.

➤ To set a substitution variable, use any of the following methods:

Tool	Topic	Location
Administration Services	Managing Substitution Variables	*Essbase Administration Services Online Help*
MaxL	**alter system** **alter application** **alter database**	*Technical Reference*
ESSCMD	CREATEVARIABLE	*Technical Reference*

Deleting Substitution Variables

You may need to delete a substitution variable that is no longer used.

➤ To delete a substitution variable, use any of the following methods:

Tool	Instructions	For More Information
Administration Services	Managing Substitution Variables	*Essbase Administration Services Online Help*
MaxL	**alter system** **alter application** **alter database**	*Technical Reference*
ESSCMD	DELETEVARIABLE	*Technical Reference*

Updating Substitution Variables

You can modify or update existing substitution variables. Be sure to consult "Rules for Setting Substitution Variable Names and Values" on page 134 before updating a substitution variable.

➤ To update a substitution variable, use any of the following methods:

Tool	Instructions	For More Information
Administration Services	Managing Substitution Variables	*Essbase Administration Services Online Help*
MaxL	**alter system** **alter application** **alter database**	*Technical Reference*
ESSCMD	UPDATEVARIABLE	*Technical Reference*

Copying Substitution Variables

You can copy substitution variables to any OLAP Server, application, or database to which you have appropriate access.

➤ To copy a substitution variable, see "Copying Substitution Variables" in *Essbase Administration Services Online Help*.

Using Location Aliases

A location alias is a descriptor for a data source. A location alias maps an alias name for a database to the location of that database. A location alias is set at the database level and specifies an alias, a server, an application, a database, a username, and a password. You need database designer permissions to set location aliases.

After you create a location alias, you can use the alias to refer to that database. If the location of the database changes, you can edit the location definition accordingly.

Note: You can use location aliases only with the @XREF function. With this function, you can retrieve a data value from another database to include in a calculation on the current database. In this case, the location alias points to the database from which the value is to be retrieved. For more information on @XREF, see the *Technical Reference*.

- "Creating Location Aliases" on page 137
- "Editing or Deleting Location Aliases" on page 137

Creating Location Aliases

You can create a location alias for a particular database.

➤ To create a location alias, use any of the following methods:

Tool	Topic	Location
Administration Services	Creating Location Aliases	*Essbase Administration Services Online Help*
MaxL	**create location alias**	*Technical Reference*
ESSCMD	CREATELOCATION	*Technical Reference*

Editing or Deleting Location Aliases

You can edit or delete location aliases that you previously created.

➤ To edit or delete a location alias, use any of the following methods:

Tool	Topic	Location
Administration Services	Editing or Deleting Location Aliases	*Essbase Administration Services Online Help*
MaxL	**display location alias** **drop location alias**	*Technical Reference*
ESSCMD	LISTLOCATIONS DELETELOCATION	*Technical Reference*

Chapter 8

Creating and Changing Database Outlines

The database outline defines the structure of the database. Outline Editor displays the dimension hierarchy of an outline visually. This chapter explains how to create and manage an Analytic Services database outline. All examples in this chapter are based on the Sample Basic database shipped with Analytic Services.

Note: The information in this chapter is designed for block storage databases. Some of the information is not relevant to aggregate storage databases. For detailed information on the differences between aggregate and block storage, see Chapter 57, "Comparison of Aggregate and Block Storage." For information on creating aggregate storage applications, see Chapter 58, "Aggregate Storage Applications, Databases, and Outlines."

This chapter contains the following sections:

You can also change outlines using data sources and rules files. For more information, see Chapter 16, "Understanding Data Loading and Dimension Building".

For basic information about outlines, see Chapter 2, "Understanding Multidimensional Databases."

For information on setting properties in an outline, see Chapter 9, "Setting Dimension and Member Properties."

Process for Creating Outlines

This section provides an overview of creating outlines using Outline Editor. For more information about outlines, see "Dimensions and Members" on page 33. To learn how to use Outline Editor, see "About Outline Editor" and "Customizing Outline Viewer and Outline Editor" in the *Essbase Administration Services Online Help*.

➤ To create an outline, follow these steps:

1. Create a new database. The new database automatically contains a blank outline. See "Creating Applications and Databases" on page 131.
2. Open the outline. See "Creating and Editing Outlines" on page 140.
3. Add dimensions and members to the outline. See "Adding Dimensions and Members to an Outline" on page 143.
4. Set each dimension as dense or sparse. See "Setting Data Storage Properties" on page 146.
5. Position dimensions and members in the outline. See "Positioning Dimensions and Members" on page 147.
6. Set dimension and member properties. See Chapter 9, "Setting Dimension and Member Properties."
7. If necessary, create attribute dimensions and associate them with the appropriate base dimensions. See Chapter 10, "Working with Attributes."
8. Verify and save the outline. See "Verifying Outlines" on page 148 and "Saving Outlines" on page 150.

Creating and Editing Outlines

When a database is created, Analytic Services creates an outline for that database automatically. The outline has the same name as the database (*dbname*`.otl`) and is stored in the database directory on Analytic Server. You can create content in the new outline in the following ways:

- Open the empty outline created by default when you create a database and add content manually.
- Copy an existing outline to the current database and change the existing outline.
- Create content in the outline using data sources and rules files. For more information, see Chapter 16, “Understanding Data Loading and Dimension Building”.

In Administration Services, you can open an existing outline in edit mode (using Outline Editor) or in read-only mode (using Outline Viewer). Outlines opened in edit mode consume more memory on the Administration Server than outlines opened in read-only mode. For more information, see “Opening and Editing Outlines” in *Essbase Administration Services Online Help*.

When you open an outline in Outline Editor, you can view and manipulate the dimensions and members graphically. An outline is always locked when it is opened in edit mode. If you have Supervisor permissions, you can unlock a locked outline. For more information, see “Locking and Unlocking Outlines” in *Essbase Administration Services Online Help*.

CAUTION: If you open the same outline with two instances of the Administration Services Console using the same login ID, each save overwrites the changes of the other instance. Because it can be difficult to keep track of what changes are saved or overwritten, Hyperion does not recommend this practice.

➤ To create a new outline or open an existing outline, use any of the following methods:

Tool	Topic	Location
Administration Services	Opening and Editing Outlines	*Essbase Administration Services Online Help*
MaxL	**create database**	*Technical Reference*
ESSCMD	CREATEDB	*Technical Reference*

➤ To copy an existing outline, use any of the following methods:

Tool	Topic	Location
Administration Services	Copying Outlines	*Essbase Administration Services Online Help*
MaxL	**create database as**	*Technical Reference*
ESSCMD	COPYDB	*Technical Reference*

Locking and Unlocking Outlines

In Outline Editor, an outline is always locked when it is opened in edit mode. Analytic Services unlocks the outline when the outline is closed. When an outline is locked, Analytic Services does not allow other users to save over, rename, delete, or edit the outline. When you attempt to edit a locked outline, you are given an option to view the outline in Outline Viewer. For more information about Outline Editor and Outline Viewer, see "Creating and Editing Outlines" on page 140.

If you have Supervisor permissions, you can unlock a locked outline. Before you forcefully unlock a locked outline, make sure that no one else is working with it.

Note: Analytic Services uses a different process for locking and unlocking outlines than for other database objects. For more information about object locking, see "Locking and Unlocking Objects" on page 964.

➤ To unlock an outline, use any of the following methods:

Tool	Topic	Location
Administration Services	Locking and Unlocking Outlines	*Essbase Administration Services Online Help*
MaxL	**create database as**	*Technical Reference*
ESSCMD	UNLOCKOBJECT	*Technical Reference*

Adding Dimensions and Members to an Outline

After you create an outline, you can add dimensions and member hierarchies to the outline manually using Outline Editor or with a data source and rules file using Data Prep Editor.

Be sure to consult "Understanding the Rules for Naming Dimensions and Members" on page 143 before naming dimensions and members.

- To add dimensions and members to an outline using Outline Editor, see "Adding Dimensions to Outlines" and "Adding Members to Dimensions" in the *Essbase Administration Services Online Help*.
- To add dimensions and members to an outline using Data Prep Editor, see "Creating a Dimension Build Rules File" in the *Essbase Administration Services Online Help*.
- To add dimensions and members dynamically (using a rules file) from Outline Editor, see "Updating an Outline Dynamically Using a Rules File" in the *Essbase Administration Services Online Help*.

Understanding the Rules for Naming Dimensions and Members

When naming dimensions, members, and aliases in the database outline, follow these rules:

- Use no more than the maximum lengths that are specified in Appendix A, "Limits".
- Names are not case-sensitive unless case-sensitivity is enabled. See "Setting Outline Properties" in the *Essbase Administration Services Online Help*.
- Do not use " (quotation marks) or tabs anywhere in a name.

- Do not use the following characters at the beginning of a name:

@	at signs	()	parentheses
\	backslashes	.	periods
{ }	braces	+	plus signs
,	commas	'	single quotation marks
-	dashes, hyphens, or minus	_	underscores
=	equal signs	|	vertical bars
<	less than signs		

- Do not place spaces at the beginning or end of a name. Analytic Services ignores spaces at the beginning or end of a name.
- Do not use the following words as dimension or member names:
  - Calculation script commands, operators, and keywords. For a list of commands, see the *Technical Reference*.
  - Report writer commands. For a list of commands, see the *Technical Reference*.
  - Function names and function arguments. For a list of functions, see the *Technical Reference*.
  - Names of other dimensions, members (unless the member is shared), generation names, level names, and aliases in the database.

– Any of the following words:

ALL	GENRANGE	OR
AND	GROUP	PAREN
ASSIGN	GT	PARENPARM
CALC	ID	PERCENT
CALCMBR	IDERROR	PLUS
COPYFORWARD	INTEGER	RELOP
CROSSDIM	LE	SET
CURMBRNAME	LEVELRANGE	SKIPBOTH
DIM	LOOPBLOCK	SKIPMISSING
DIMNAME	LOOPPARMS	SKIPNONE
DIV	LT	SKIPZERO
DYNAMIC	MBR	TO
EMPTYPARM	MBRNAME	TOLOCALRATE
EQ	MBRONLY	TRAILMISSING
EQOP	MINUS	TRAILSUM
EXCEPT	MISSING	UMINUS
EXP	MUL	UPPER
EXPERROR	MULOP	VARORXMBR
FLOAT	NE	XMBRONLY
FUNCTION	NON	$$$UNIVERSE$$$
GE	NONINPUT	#MISSING
GEN	NOT	#MI

Note: If you enable Dynamic Time Series members, do not use the associated generation names—History, Year, Season, Period, Quarter, Month, Week, or Day. See "Applying Predefined Generation Names to Dynamic Time Series Members" on page 576.

- In calculation scripts, report scripts, filter definitions, partition definitions, or formulas, you must enclose member names in quotation marks ("") in the following situations:
  - The name starts with one or more numerals (for example, 100).
  - The name contains spaces or any of the following characters:

&	ampersand	!	exclamation point
*	asterisk	>	greater than sign
@	at sign	<	less than sign

\	backslash	()	parentheses
{ }	braces	%	percent sign
[ ]	brackets	.	period
:	colon	+	plus sign
,	comma	;	semicolon
-	dash, hyphen, or minus	/	slash
=	equal sign	~	tilde

- In calculation scripts and formulas, you must enclose the following member names in quotation marks (""):

  BEGIN
  DOUBLE
  END
  MACRO
  MEMBER
  RANGE
  STRING
  THEN

Tips to make names unique:

- Concatenate the member and alias names; for example, 100-10_Cola and 100-10_Smith.
- Add prefixes or suffixes to member names. For example, if the parent is the state and several states have a city called Jackson, appending the state abbreviation creates the unique names Jackson_CA, Jackson_IL, and Jackson_MI.

Setting Data Storage Properties

When you create new dimensions and save an outline, Analytic Services automatically sets the new dimensions in the outline as sparse. You can change the dimension storage type according to the optimal configuration for the database.

For information about choosing dense or sparse storage, see "Selection of Sparse and Dense Dimensions" on page 68. You must set a standard dimension with which you plan to associate an attribute dimension as sparse.

➤ To set data storage properties using Outline Editor, see "Setting Dimensions as Dense or Sparse" in the *Essbase Administration Services Online Help*.

Positioning Dimensions and Members

Dimensions are the highest level of organization in an outline. Dimensions contain members. You can nest members inside of other members in a hierarchy. For more information on dimensions and members, see "Dimensions and Members" on page 33.

The following sections describe how to position dimensions and members in the outline:

- "Moving Dimensions and Members" on page 147
- "Sorting Dimensions and Members" on page 148

Note: The relative locations of dimensions in an outline can affect calculation and retrieval performance times. See "Designing an Outline to Optimize Performance" on page 100.

Moving Dimensions and Members

After you create dimensions and members, you can rearrange them within the outline. Before moving members and dimensions in an outline consider the following information:

- The positions of dimensions and members in an outline can affect performance. See "Optimizing Outline Performance" on page 195.
- Moving dimensions and members can affect the performance of calculations and retrievals. For information about performance for calculation and retrieval, see "Designing an Outline to Optimize Performance" on page 100.
- Moving members could move a shared member before the actual member in the outline, something that Hyperion does not recommend.
- If you add, delete, or move non-attribute dimensions or members, Analytic Services restructures the database, and you must recalculate the data.
- You must position attribute dimensions at the end of the outline. If you do not position attribute dimensions at the end of the outline, during outline verification, Analytic Services prompts you to move them there.

➤ To position dimensions and members using Outline Editor, see “Manipulating Dimensions and Members in an Outline” in the *Essbase Administration Services Online Help*.

Sorting Dimensions and Members

You can have Analytic Services arrange dimensions within an outline or members within a dimension in alphabetical order (A to Z) or reverse alphabetical order (Z to A). For a list of consequences of sorting dimensions and members, see “Moving Dimensions and Members” on page 147.

When you sort level 0 members of numeric attribute dimensions in outlines, the members are sorted by their values. For example, Figure 47 shows text and numeric versions of the Sizes attribute dimension after sorting the members in ascending order. The members of the numeric attribute dimension are sequenced by the numeric values of the members; the member 8 is before the other members. In the text attribute dimension, because the characters are sorted left to right, the member 8 is after the member 24.

Figure 47: Sorting Numeric Versus Text Attribute Dimension in Ascending Order

Sizes Attribute (Type: Text)	Sizes Attribute (Type: Numeric)
Ounces	Ounces
12	8
16	12
24	16
8	24

You cannot sort Boolean attribute dimensions. For more information about attribute dimension types, see “Understanding Attribute Types” on page 187.

➤ To sort members using Outline Editor, see “Sorting Members” in the *Essbase Administration Services Online Help*.

Verifying Outlines

You can verify an outline automatically when you save it or you can verify the outline manually at any time. When verifying an outline, Analytic Services checks the following items:

- All member and alias names are valid. Members and aliases cannot have the same name as other members, aliases, generations, or levels. See “Understanding the Rules for Naming Dimensions and Members” on page 143 for more information.

- Only one dimension is tagged as accounts, time, currency type, or country.
- Shared members are valid as described in "Understanding the Rules for Shared Members" on page 165.
- Level 0 members are not tagged as label only.
- Label-only members have not been assigned formulas.
- The currency category and currency name are valid for the currency outline.
- Dynamic Calc members in sparse dimensions do not have more than 100 children.
- If a parent member has one child and if that child is a Dynamic Calc member, the parent member must also be Dynamic Calc.
- If a parent member has one child and if that child is a Dynamic Calc, Two-Pass member, the parent member must also be Dynamic Calc, Two-Pass.
- The two names of members of Boolean attribute dimensions are the same as the two Boolean attribute dimension member names defined for the outline.
- The level 0 member name of a date attribute dimension must match the date format name setting (mm-dd-yyyy or dd-mm-yyyy). If the dimension has no members, because the dimension name is the level 0 member, the dimension name must match the setting.
- The level 0 member name of a numeric attribute dimension is a numeric value. If the dimension has no members, because the dimension name is the level 0 member, the dimension name must be a numeric value.
- Attribute dimensions are located at the end of the outline, following all standard dimensions.
- Level 0 Dynamic Calc members of standard dimensions have a formula.
- Formulas for members are valid.
- In a Hybrid Analysis outline, only the level 0 members of a dimension can be Hybrid Analysis-enabled.

During outline verify, Analytic Services also performs the following conversions to appropriate numeric attribute dimension member names and displays them in the outline:

- It moves minus signs in member names from the front to the end of the name; for example, -1 becomes 1-.
- It strips out leading or trailing zeroes in member names; for example, 1.0 becomes 1, and 00.1 becomes 0.1.

For more information about numeric attribute dimensions, see "Understanding Attribute Types" on page 187.

➤ To verify an outline, see "Verifying Outlines" in the *Essbase Administration Services Online Help*.

Saving Outlines

You can save outlines to the Analytic Server or to a client computer or network. By default, Analytic Services saves outlines to the database directory on Analytic Server. If you are saving changes to an existing outline, Analytic Services may restructure the outline. For example, if you change a member name from Market to Region, Analytic Services moves data stored in reference to Market to Region. Each time that you save an outline, Analytic Services verifies the outline to make sure that it is correct.

➤ To save an outline, see "Saving Outlines" in the *Essbase Administration Services Online Help*.

For more information about adding or deleting members before saving an outline, see the following sections:

- "Saving an Outline with Added Standard Dimensions" on page 151
- "Saving an Outline with One or More Deleted Standard Dimensions" on page 151
- "Creating Sub-Databases Using Deleted Members" on page 151

Saving an Outline with Added Standard Dimensions

If you add one or more new standard (non-attribute) dimensions, then any data that existed previously in the database must be mapped to a member of each new dimension. For example, adding a dimension called Channel to the Sample Basic outline implies that all previous data in Sample Basic is associated with a particular channel or the sum of all channels.

If you add one or more new standard dimensions and then attempt to save the outline, Analytic Services prompts you to associate data of previously existing dimensions to one member of each new dimension.

Saving an Outline with One or More Deleted Standard Dimensions

If you delete one or more standard (non-attribute) dimensions, the data associated with only one member of each deleted dimension can be retained. For example, removing a dimension called Market from the outline implies that all of the data that remains in the database after the restructure operation is associated with a single, specified member of the Market dimension.

If you delete one or more dimensions and then attempt to save the outline, Analytic Services prompts you to select a member of the deleted dimension whose data values will be retained and associated with the members of the other dimensions.

If you delete an attribute dimension, Analytic Services deletes the associations to its base dimension. See Chapter 10, “Working with Attributes.”

Creating Sub-Databases Using Deleted Members

➤ To create a sub-database:

1. Delete a dimension from an existing outline.
2. Save the database using a different name, and specify the member to keep. Only one member can be kept when a dimension is deleted. For more information, see “Saving an Outline with One or More Deleted Standard Dimensions” on page 151.

Chapter 9

Setting Dimension and Member Properties

After you create and organize the outline, as described in Chapter 8, "Creating and Changing Database Outlines," you are ready to specify how the dimensions and members in the outline behave. This chapter describes dimension and member properties and how to set properties.

Note: The information in this chapter is designed for block storage databases. Some of the information is not relevant to aggregate storage databases. For detailed information on the differences between aggregate and block storage, see Chapter 57, "Comparison of Aggregate and Block Storage." For information on creating aggregate storage applications, see Chapter 58, "Aggregate Storage Applications, Databases, and Outlines."

This chapter contains the following topics:

Setting Dimension Types

When you tag a dimension as a specific type, the dimension can access built-in functionality designed for that type. For example, if you define a dimension as accounts, you can specify accounting measures for members in that dimension. Analytic Services calculates the two primary dimension types, time and accounts, before other dimensions in the database. By default, all dimensions are tagged as none.

The following sections describe the different dimension types:

- "Creating a Time Dimension" on page 154
- "Creating an Accounts Dimension" on page 155
- "Creating a Country Dimension" on page 159
- "Creating Currency Partitions" on page 159
- "Creating Attribute Dimensions" on page 160

➤ To set a dimension type, see "Setting the Dimension Type" in the *Essbase Administration Services Online Help*.

Creating a Time Dimension

Tag a dimension as time if it contains members that describe how often you collect and update data. In the Sample Basic database, for example, the Year dimension is tagged as time, as are its descendants—all Qtr members and the months (such as Jan). The time dimension also enables several accounts dimension functions, such as first and last time balances.

Follow these rules when tagging a dimension as time:

- You can tag only one dimension in an outline as time.
- All members in the time dimension inherit the time property.
- You can add time members to dimensions that are not tagged as time.
- You can create an outline that does not have a time dimension.

➤ To tag a dimension as time, see "Tagging a Time Dimension" in the *Essbase Administration Services Online Help*.

Creating an Accounts Dimension

Tag a dimension as accounts if it contains items that you want to measure, such as profit or inventory.

Follow these rules when tagging a dimension as accounts:

- You can tag only one dimension in an outline as accounts.
- You can create an outline that does not have an accounts dimension.
- All members in the accounts dimension inherit the accounts property.
- You can specify that members of the accounts dimension are calculated on the second pass through an outline. For more information, see "Setting Two-Pass Calculations" on page 174.

➤ To tag a dimension as accounts, see "Tagging an Accounts Dimension" in the *Essbase Administration Services Online Help*.

The following sections describe built-in functionality for accounts dimensions:

Setting Time Balance Properties

If a member of the accounts dimension uses the time balance property, it affects how Analytic Services calculates the parent of that member in the time dimension. By default, a parent in the time dimension is calculated based on the consolidation and formulas of its children. For example, in the Sample Basic database, the Qtr1 member is the sum of its children (Jan, Feb, and Mar). However, setting a time balance property causes parents, for example Qtr1, to roll up differently.

➤ To set time balance properties, see "Setting Time Balance Properties" in the *Essbase Administration Services Online Help*.

Example of Time Balance as None

None is the default value. When you set the time balance property as none, Analytic Services rolls up parents in the time dimension in the usual way—the value of the parent is based on the formulas and consolidation properties of its children.

Example of Time Balance as First

Set the time balance as first when you want the parent value to represent the value of the first member in the branch (often at the beginning of a time period).

For example, assume that you have a member named OpeningInventory that represents the inventory at the beginning of the time period. If the time period was Qtr1, then OpeningInventory represents the inventory at the beginning of Jan; that is, the OpeningInventory for Qtr1 is the same as the OpeningInventory for Jan. For example, if you had 50 cases of Cola at the beginning of Jan, you also had 50 cases of Cola at the beginning of Qtr1.

To accomplish this task, tag OpeningInventory as first. Figure 48 shows this sample consolidation.

Figure 48: Consolidation of OpeningInventory Tagged as First

```
OpeningInventory (TB First), Cola, East, Actual, Jan(+), 50
OpeningInventory (TB First), Cola, East, Actual, Feb(+), 60
OpeningInventory (TB First), Cola, East, Actual, Mar(+), 70
OpeningInventory (TB First), Cola, East, Actual, Qtr1(+), 50
```

Example of Time Balance as Last

Set the time balance as last when you want the parent value to represent the value of the last member in the branch (often at the end of a time period).

For example, assume that you have a member named EndingInventory that represents the inventory at the end of the time period. If the time period was Qtr1, then EndingInventory represents the inventory at the end of Mar; that is, the EndingInventory for Qtr1 is the same as the EndingInventory for Mar. For example, if you had 70 cases of Cola at the end of Mar, you also had 70 cases of Cola at the end of Qtr1.

To accomplish this task, tag EndingInventory as last. Figure 49 shows this sample consolidation.

Figure 49: Consolidation of EndingInventory Tagged as Last

```
EndingInventory (TB Last), Cola, East, Actual, Jan(+), 50
EndingInventory (TB Last), Cola, East, Actual, Feb(+), 60
EndingInventory (TB Last), Cola, East, Actual, Mar(+), 70
EndingInventory (TB Last), Cola, East, Actual, Qtr1(+), 70
```

Example of Time Balance as Average

Set the time balance as average when you want the parent value to represent the average value of its children.

For example, assume that you have a member named AverageInventory that represents the average of the inventory for the time period. If the time period was Qtr1, then AverageInventory represents the average of the inventory during Jan, Feb, and Mar.

To accomplish this task, tag AverageInventory as average. Figure 50 shows this sample consolidation.

Figure 50: Consolidation of AverageInventory Tagged as Average

```
AverageInventory (TB Average), Cola, East, Actual, Jan(+), 60
AverageInventory (TB Average), Cola, East, Actual, Feb(+), 62
AverageInventory (TB Average), Cola, East, Actual, Mar(+), 67
AverageInventory (TB Average), Cola, East, Actual, Qtr1(+), 63
```

Setting Skip Properties

If you set the time balance as first, last, or average, you must set the skip property to tell Analytic Services what to do when it encounters missing values or values of 0.

The following table describes how each setting determines what Analytic Services does when it encounters a missing or zero value.

Setting	Action Analytic Services Takes
None	Does not skip data when calculating the parent value.
Missing	Skips #MISSING data when calculating the parent value.

Setting	Action Analytic Services Takes
Zeros	Skips data that equals zero when calculating the parent value.
Missing and Zeros	Skips both #MISSING data and data that equals zero when calculating the parent value.

If you mark a member as last with a skip property of missing or missing and zeros, then the parent of that time period matches the last non-missing child. In Figure 51, for example, EndingInventory is based on the value for Feb, because Mar does not have a value.

Figure 51: Example of Skip Property

```
Cola, East, Actual, Jan, EndingInventory (Last), 60
Cola, East, Actual, Feb, EndingInventory (Last), 70
Cola, East, Actual, Mar, EndingInventory (Last), #MI
Cola, East, Actual, Qtr1, EndingInventory (Last), 70
```

Setting Variance Reporting Properties

Variance reporting properties determine how Analytic Services calculates the difference between actual and budget data in a member with the @VAR or @VARPER function in its member formula. Any member that represents an expense to the company requires an expense property.

When you are budgeting *expenses* for a time period, the actual expenses should be lower than the budget. When actual expenses are greater than budget, the variance is negative. The @VAR function calculates Budget – Actual. For example, if budgeted expenses were $100, and you actually spent $110, the variance is -10.

When you are budgeting *non-expense* items, such as sales, the actual sales should be higher than the budget. When actual sales are less than budget, the variance is negative. The @VAR function calculates Actual - Budget. For example, if budgeted sales were $100, and you actually made $110 in sales, the variance is 10.

By default, members are non-expense.

➤ To set variance reporting properties, see “Setting Variance Reporting Properties” in the *Essbase Administration Services Online Help*.

Setting Analytic Services Currency Conversion Properties

Currency conversion properties define categories of currency exchange rates. These properties are used only in currency databases on members of accounts dimensions. For a comprehensive discussion of currency conversion, see Chapter 12, "Designing and Building Currency Conversion Applications."

➤ To set currency conversion properties, see "Assigning Currency Categories to Accounts Members" in the *Essbase Administration Services Online Help*.

Creating a Country Dimension

Use country dimensions to track business activities in multiple countries. If you track business activity in the United States and Canada, for example, the country dimension should contain states, provinces, and countries. If a dimension is tagged as country, you can set the currency name property. The currency name property defines what type of currency this market region uses.

In a country dimension, you can specify the type of currency used in each member. For example, in the Interntl application and database shipped with Analytic Services, Canada has three markets—Vancouver, Toronto, and Montreal. They use the same currency, Canadian dollars.

This dimension type is used for currency conversion applications. For a comprehensive discussion of currency conversion, see Chapter 12, "Designing and Building Currency Conversion Applications."

➤ To tag a dimension as country, see "Tagging a Country Dimension" in the *Essbase Administration Services Online Help*.

Creating Currency Partitions

Use currency partition members to separate local currency members from a base currency defined in the application. If the base currency for analysis is US dollars, for example, the local currency members would contain values based on the currency type of the region, such as Canadian dollars.

This dimension type is used for currency conversion applications. For information about how to create and use currency partitions, see Chapter 12, "Designing and Building Currency Conversion Applications."

For complete information about how to design and implement currency applications, see Chapter 12, “Designing and Building Currency Conversion Applications.”

➤ To tag a dimension as currency partition, see “Creating a Currency Partition” in the *Essbase Administration Services Online Help*.

Creating Attribute Dimensions

Use attribute dimensions to report and aggregate data based on characteristics of standard dimensions. In the Sample Basic database, for example, the Product dimension is associated with the Ounces attribute dimension. Members of the Ounces attribute dimension categorize products based on their size in ounces.

Be sure to review the rules for using attribute dimensions in Chapter 10, “Working with Attributes.”

➤ To tag a dimension as attribute, see “Tagging an Attribute Dimension” in the *Essbase Administration Services Online Help*.

Setting Member Consolidation

Member consolidation properties determine how children roll up into their parents. By default, new members are given the addition (+) operator, meaning that members are added. For example, Jan, Feb, and Mar figures are added and the result stored in their parent, Qtr1.

Note: Analytic Services does not use consolidation properties with members of attribute dimensions. See “Calculating Attribute Data” on page 200 for an explanation of how the attribute dimensions works.

Table 10 describes each operator.

Table 10: Consolidation Operators

Operator	Description
+	Adds the member to the result of previous calculations performed on other members. + is the default operator.
-	Multiplies the member by -1 and then adds it to the sum of previous calculations performed on other members.
*	Multiplies the member by the result of previous calculations performed on other members.
/	Divides the member into the result of previous calculations performed on other members.
%	Divides the member into the sum of previous calculations performed on other members. The result is multiplied by 100 to yield a percentage value.
~	Does not use the member in the consolidation to its parent.

➤ To set member consolidation properties, see "Setting Member Consolidation Properties" in the *Essbase Administration Services Online Help*.

Calculating Members with Different Operators

When siblings have different operators, Analytic Services calculates the data in top-down order. The following section describes how Analytic Services calculates the members in Figure 52.

Figure 52: Sample Roll Up

```
Parent1
   Member1 (+)   10
   Member2 (+)   20
   Member3 (-)   25
   Member4 (*)   40
   Member5 (%)   50
   Member6 (/)   60
   Member7 (~)   70
```

Analytic Services calculates Member1 through Member4 in Figure 52 as follows:

Figure 53: Sample Roll Up for Members 1 through 4

```
(((Member1 + Member2) + (-1)Member3) * Member4) = X
(((10 + 20) + (-25)) * 40) = 200
```

If the result of Figure 53 is X, then Member5 consolidates as follows:

Figure 54: Sample Roll Up for Member 5

```
(X/Member5) * 100 = Y
(200/50) * 100 = 400
```

If the result of Figure 54 is Y, then Member6 consolidates as follows:

Figure 55: Sample Roll Up for Member 6

```
Y/Member6 = Z
400/60 = 66.67
```

Because it is set to No Consolidation(~), Analytic Services ignores Member7 in the consolidation.

Determining How Members Store Data Values

You can determine how and when Analytic Services stores the data values for a member. For example, you can tell Analytic Services to only calculate the value for a member when a user requests it and then discard the data value. Table 11 describes each storage property.

Table 11: Choosing Storage Properties

Storage Property	When to Use	For More Information
Store	Store the data value with the member.	"Understanding Stored Members" on page 163
Dynamic Calc and Store	Not calculate the data value until a user requests it, and then store the data value.	"Understanding Dynamic Calculation Members" on page 163
Dynamic Calc	Not calculate the data value until a user requests it, and then discard the data value.	"Understanding Dynamic Calculation Members" on page 163

Table 11: Choosing Storage Properties (Continued)

Storage Property	When to Use	For More Information
Never share	Not allow members to be shared implicitly. Members tagged as Never share can only be explicitly shared. To explicitly share a member, create the shared member with the same name and tag it as shared.	"Understanding Implied Sharing" on page 168
Label only	Create members for navigation only, that is, members that contain no data values.	"Understanding Label Only Members" on page 164
Shared member	Share values between members. For example, in the Sample Basic database, the 100-20 member is stored under the 100 parent and shared under Diet parent.	"Understanding Shared Members" on page 164

➤ To set member storage properties, see "Setting Member Storage Properties" in the *Essbase Administration Services Online Help*.

Understanding Stored Members

Stored members contain calculated values that are stored with the member in the database after calculation. By default, members are set as stored.

➤ To define a member as stored, see "Setting Member Storage Properties" in the *Essbase Administration Services Online Help*.

Understanding Dynamic Calculation Members

When a member is Dynamic Calc, Analytic Services does not calculate the value for that member until a user requests it. After the user views it, Analytic Services does not store the value for that member. If you tag a member as Dynamic Calc and Store, Analytic Services performs the same operation as for a Dynamic Calc member, except that Analytic Services stores the data value for that member after the user views it.

For a comprehensive discussion of Dynamic Calc and Dynamic Calc and Store members, see Chapter 25, "Dynamically Calculating Data Values."

Analytic Services automatically tags members of attribute dimensions as Dynamic Calc. You cannot change this setting.

➤ To tag a member as Dynamic Calc, see "Setting Member Storage Properties" in the *Essbase Administration Services Online Help*.

Understanding Label Only Members

Label only members have no data associated with them. Use them to group members or to ease navigation and reporting from the Spreadsheet Add-in. Typically, you should give label only members the "no consolidation" property. For more information about member consolidation, see "Setting Member Consolidation" on page 160.

You cannot associate attributes with label only members. If you tag as label only a base dimension member that has attributes associated with it, Analytic Services removes the attribute associations and displays a warning message.

➤ To tag a member as label only, see "Setting Member Storage Properties" in the *Essbase Administration Services Online Help*.

Understanding Shared Members

The data values associated with a shared member come from another member with the same name. The shared member stores a pointer to data contained in the other member and the data is only stored once. To define a member as shared, there must be an actual non-shared member of the same name. For example, in the Sample Basic database, the 100-20 member under 100 stores the data for that member. The 100-20 member under Diet points to that value.

Shared members are typically used to calculate the same member across multiple parents. For example, you might want to calculate a Diet Cola member in both the 100 and Diet parents.

Using shared members lets you use members repeatedly throughout a dimension. Analytic Services stores the data value only once, but it displays in multiple locations. Storing the data value only once offers considerable space saving as well as processing efficiency.

➤ To tag a member as shared, see "Setting Member Storage Properties" in the *Essbase Administration Services Online Help*.

Use these sections to learn more about shared members:

- "Understanding the Rules for Shared Members" on page 165
- "Understanding Shared Member Retrieval During Drill-Down" on page 166
- "Understanding Implied Sharing" on page 168

Understanding the Rules for Shared Members

Follow these rules when creating shared members:

- The shared members must be in the same dimension. For example, both 100-20 members in the Sample Basic database are in the Product dimension.
- Shared members cannot have children.
- You can have an unlimited number of shared members with the same name.
- You cannot assign UDAs or formulas to shared members.
- You cannot associate attributes with shared members.
- If you assign accounts properties to shared members, the values for those accounts properties are taken from the base member, even if you change the accounts properties on the shared member.
- You can assign aliases to shared members.
- You should not create an outline where shared members are located before actual members in a dimension.
- Avoid creating complex relationships between real and shared members that will be part of an attribute calculation, or your calculation may return unexpected results. For details, see "Understanding Attribute Calculation and Shared Members" on page 208.

Understanding Shared Member Retrieval During Drill-Down

Analytic Services retrieves shared members during drill-down, depending on their location in the spreadsheet. Analytic Services follows three rules during this type of retrieval:

- Analytic Services retrieves stored members (not their shared member counterparts) by default.
- Analytic Services retrieves from the bottom of a spreadsheet first.
- If the parent of a shared member is a sibling of the stored member counterpart of one of the shared members, Analytic Services retrieves the stored member.

Example of Shared Members from a Single Dimension

If you created a test dimension with all shared members based on the members of the dimension East from the Sample Basic outline, the outline would be similar to the following:

```
Database:  (Current Alias Table: Default)
    Year Time (Active Dynamic Time Series Members: H-T-D, Q-T-D) (Dynamic Calc)
    Measures Accounts (Label Only)
    Product {Caffeinated, Ounces, Pkg Type, Intro Date }
    Market {Population }
        East (+) (UDAs: Major Market)
            New York (+) (UDAs: Major Market) {Population:21000000}
            Massachusetts (+) (UDAs: Major Market) {Population:9000000}
            Florida (+) (UDAs: Major Market) {Population:15000000}
            Connecticut (+) (UDAs: Small Market) {Population:6000000}
            New Hampshire (+) (UDAs: Small Market) {Population:3000000}
        West (+)
        South (+) (UDAs: Small Market)
        Central (+) (UDAs: Major Market)
        test (+)
            New York (+) (Shared Member)
            Massachusetts (+) (Shared Member)
            Florida (+) (Shared Member)
            Connecticut (+) (Shared Member)
            New Hampshire (+) (Shared Member)
    Scenario (Label Only)
```

If you retrieved just the children of East, all results would be from stored members because Analytic Services retrieves stored members by default.

If, however, you retrieved data with the children of test above it in the spreadsheet, Analytic Services would retrieve the shared members:

New York
Massachusetts
Florida
Connecticut
New Hampshire
test

If you moved test above its last two children, Analytic Services would retrieve the first three children as shared members, but the last two as stored members. Similarly, if you inserted a member in the middle of the list above which was not a sibling of the shared members (for example, California inserted between Florida and Connecticut), then Analytic Services would retrieve shared members only between the non-sibling and the parent (in this case, between California and test).

Example of Retrieval with Crossed Generation Shared Members

You could modify the Sample Basic outline to create a shared member whose stored member counterpart was a sibling to its own parent:

```
Database:  (Current Alias Table: Default)
    Year Time (Active Dynamic Time Series Members: H-T-D, Q-T-D) (Dynamic Calc)
    Measures Accounts (Label Only)
    Product {Caffeinated, Ounces, Pkg Type, Intro Date }
    Market {Population }
        East (+) (UDAs: Major Market)
            New York (+) (UDAs: Major Market) {Population:21000000}
            Massachusetts (+) (UDAs: Major Market) {Population:9000000}
            Florida (+) (UDAs: Major Market) {Population:15000000}
            Connecticut (+) (UDAs: Small Market) {Population:6000000}
            New Hampshire (+) (UDAs: Small Market) {Population:3000000}
        West (+)
        South (+) (UDAs: Small Market)
        Central (+) (UDAs: Major Market)
        test (+)
            west (+) (Shared Member)
            New York (+) (Shared Member)
            Massachusetts (+) (Shared Member)
            Florida (+) (Shared Member)
            Connecticut (+) (Shared Member)
            New Hampshire (+) (Shared Member)
```

If you created a spreadsheet with shared members in this order, Analytic Services would retrieve all the shared members, except it would retrieve the stored member West, not the shared member west:

west
New York
Massachusetts
Connecticut
New Hampshire
test

Analytic Services retrieves the members in this order because test is a parent of west and a sibling of west's stored member counterpart, West.

Understanding Implied Sharing

The shared member property defines a shared data relationship explicitly. Some members are shared even if you do not explicitly set them as shared. These members are said to be *implied shared members*.

Analytic Services assumes (or implies) a shared member relationship in the following situations:

- **A parent has only one child.** In this situation, the parent and the child contain the same data. Analytic Services ignores the consolidation property on the child and stores the data only once—thus the parent has an implied shared relationship with the child. In Figure 56, for example, the parent 500 has only one child, 500-10, so the parent shares the value of that child.

  Figure 56: Implied Sharing of a Parent with One Child

  ```
  500 (+)
      500-10 (+)
  ```

- **A parent has only one child that consolidates to the parent.** If the parent has four children, but three of them are marked as no consolidation, then the parent and child that consolidates contain the same data. Analytic Services ignores the consolidation property on the child and stores the data only once—thus the parent has an implied shared relationship with the child. In Figure 57, for example, the parent 500 has only one child, 500-10, that rolls up to it. The

other children are marked as No Consolidate(~), so the parent implicitly shares the value of 500-10.

Figure 57: Implied Sharing of a Parent with Multiple Children

```
500 (+)
    500-10 (+)
    500-20 (~)
    500-30 (~)
```

If you do not want a member to be shared implicitly, mark the parent as Never Share so that the data is duplicated, and is not shared. See "Understanding Shared Members" on page 164 for an explanation of how shared members work.

Setting Aliases

An *alias* is an alternate name for a member or shared member. For example, members in the Product dimension in the Sample Basic database are identified both by product codes, such as 100, and by more descriptive aliases, such as Cola. Alias are stored in alias tables. Aliases can improve the readability of an outline or a report.

You can set more than one alias for a member using alias tables. For example, you could use different aliases for different kinds of reports—users may be familiar with 100-10 as Cola, but advertisers and executives may be familiar with it as The Best Cola. This list shows some products in the Sample Basic database that have two descriptive alias names:

```
Product  Default              Long Names
100-10   Cola                 The Best Cola
100-20   Diet Cola            Diet Cola with Honey
100-30   Caffeine Free Cola   All the Cola,
                                none of the Caffeine
```

For a comprehensive discussion of alias tables, see "Alias Tables" on page 170.

The following sections describe aliases:

- "Alias Tables" on page 170
- "Creating Aliases" on page 170
- "Creating and Managing Alias Tables" on page 171

Analytic Services does not support aliases for Hybrid Analysis-enabled members.

Alias Tables

Aliases are stored in one or more tables as part of a database outline. An alias table maps a specific, named set of alias names to member names. When you create a database outline, Analytic Services creates an empty alias table named Default. If you don't create any other alias tables, the aliases that you create are stored in the Default alias table.

If you want to create more than one set of aliases for outline members, create a new alias table for each set. When you view the outline or retrieve data, you can use the alias table name to indicate which set of alias names you want to see. Identifying which alias table contains the names that you want to see while viewing an outline is called making an alias table the active alias table. See "Setting an Alias Table as Active" on page 171 for further information.

For Unicode-mode applications, setting up a separate alias table for each user language enables different users to view member names in their own language. For additional information about the relevance of alias tables and Unicode, see "About the Analytic Services Implementation of Unicode" on page 884.

Creating Aliases

You can provide an alias for any member. Alias names must follow the same rules as member names. See "Understanding the Rules for Naming Dimensions and Members" on page 143.

You can use any of the following methods to create aliases in an existing alias table:

- To manually assign an alias to a member while editing an outline, see "Creating Aliases for Dimensions and Members" in the *Essbase Administration Services Online Help*.
- To use dimension build and a data source to add aliases to an alias table, see "Defining a Rules File for Adding Aliases" in the *Essbase Administration Services Online Help*.
- To import alias values from an alias table source file created in a pre-defined format, see "Importing and Exporting Alias Tables" on page 173.

Creating and Managing Alias Tables

Named alias tables enable you to display different aliases in different situations. For general information about alias tables, see "Alias Tables" on page 170. While working with alias tables, you can perform the following actions:

- "Creating a New Alias Table" on page 171
- "Setting an Alias Table as Active" on page 171
- "Copying an Alias Table" on page 172
- "Renaming an Alias Table" on page 172
- "Clearing and Deleting Alias Tables" on page 173
- "Importing and Exporting Alias Tables" on page 173

Creating a New Alias Table

An alias table contains a list of aliases to use for members in the outline.The following restrictions apply to alias tables:

- You can create up to 10 alias tables for an outline.
- The naming conventions for alias table names are the same as those for dimensions. See "Understanding the Rules for Naming Dimensions and Members" on page 143.
- Name-length restrictions depend on the Unicode-related mode of the application; see Appendix A, "Limits".
- Once an alias table is created, you cannot change its name.

➤ To create a new alias table, see "Creating Alias Tables" in *Essbase Administration Services Online Help*.

When you first create an alias table, it is empty. For information about adding aliases to an alias table and assigning them to members, see "Creating Aliases" on page 170.

Setting an Alias Table as Active

The active alias table contains the aliases that Analytic Services currently displays in the outline.

➤ To view a list of alias tables in the outline and to set the current alias table, use any of the following methods:

Tool	Topic	Location
Administration Services	Setting the Active Alias Table for Outline Editor	*Essbase Administration Services Online Help*
MaxL	**query database** **alter database**	*Technical Reference*
ESSCMD	LISTALIASES SETALIAS	*Technical Reference*

Copying an Alias Table

➤ To copy alias tables, use any of the following methods:

Tool	Topic	Location
Administration Services	Copying Alias Tables	*Essbase Administration Services Online Help*
MaxL	**alter object**	*Technical Reference*
ESSCMD	COPYOBJECT	*Technical Reference*

Renaming an Alias Table

➤ To rename an alias table, use any of the following methods:

Tool	Topic	Location
Administration Services	Renaming Alias Tables	*Essbase Administration Services Online Help*
MaxL	**alter object**	*Technical Reference*
ESSCMD	RENAMEOBJECT	*Technical Reference*

Clearing and Deleting Alias Tables

You can delete an alias table from the outline or you can clear all the aliases from an alias table without deleting the alias table itself. To clear or delete alias tables, see “Deleting and Clearing Alias Tables” in the *Essbase Administration Services Online Help*.

Importing and Exporting Alias Tables

You can import a correctly formatted text file into Analytic Services as an alias table. Alias table import files have the extension .ALT. As shown in Figure 58, alias table import files should have the following structure:

- The first line in the file starts with $ALT_NAME. Add one or two spaces followed by the name of the alias table. If the alias table name contains a blank character, enclose the name in single quotation marks.
- The last line of the file must be $END.
- Each line between the first and the last lines contains two values that are separated by one or more spaces or tabs. The first value must be the name of an existing outline member; the second value is the alias for the member.
- Any member or alias name that contains a blank or underscore must be enclosed in double quotation marks.

Figure 58: Sample Alias Table Import File

```
$ALT_NAME  'Quarters'
Qtr1   Quarter1
Jan   January
Feb   February
Mar   March
$END
```

You can also export an alias table from the Analytic Services outline to a text file. The alias table contains all the member names with aliases and the corresponding aliases.

➤ To import or export alias tables, use any of the following methods:

Tool	Topic	Location
Administration Services	Importing Alias Tables Exporting Alias Tables	*Essbase Administration Services Online Help*
MaxL	**alter database**	*Technical Reference*
ESSCMD	LOADALIAS UNLOADALIAS	*Technical Reference*

Setting Two-Pass Calculations

By default, Analytic Services calculates outlines from the bottom up—first calculating the values for the children and then the values for the parent. Sometimes, however, the values of the children may be based on the values of the parent or the values of other members in the outline. To obtain the correct values for these members, Analytic Services must first calculate the outline and then re-calculate the members that are dependent on the calculated values of other members. The members that are calculated on the second pass through the outline are called *two-pass calculations*.

For more information on bottom-up calculations, see "Using Bottom-Up Calculation" on page 1200.

For example, to calculate the ratio between Sales and Margin, Analytic Services needs first to calculate Margin, which is a parent member based on its children, including Sales. To ensure that the ratio is calculated based on a freshly calculated Margin figure, tag the Margin % ratio member as a two-pass calculation. Analytic Services calculates the database once and then calculates the ratio member again. This calculation produces the correct result.

Even though two-pass calculation is a property that you can give to any non-attribute member, it works only on the following members:

- Members of accounts dimensions
- Dynamic Calc members
- Dynamic Calc and Store members.

If two-pass calculation is assigned to other members, Analytic Services ignores it.

➤ To tag a member as two-pass, see “Setting Two-Pass Calculation Properties” in the *Essbase Administration Services Online Help*.

Creating Formulas

You can apply formulas to standard dimensions and members. You cannot set formulas for attribute dimensions and their members. The formula determines how Analytic Services calculates the outline data. For more a comprehensive discussion about formulas, see Chapter 22, “Developing Formulas”.

➤ To add formulas to a dimension or member, see “Creating and Editing Formulas in Outlines” in the *Essbase Administration Services Online Help*.

Naming Generations and Levels

You can create names for generations and levels in an outline, such as a word or phrase that describes the generation or level. For example, you might create a generation name called Cities for all cities in the outline. For information about generations and levels, see “Dimension and Member Relationships” on page 35.

Use generation and level names in calculation scripts or report scripts wherever you need to specify either a list of member names or generation or level numbers. For example, you could limit a calculation in a calculation script to all members in a specific generation. See Chapter 27, “Developing Calculation Scripts” for information about developing calculation scripts.

You can define only one name for each generation or level. When you name generations and levels, follow the same naming rules as for members. See “Understanding the Rules for Naming Dimensions and Members” on page 143.

➤ To name generations and levels using Outline Editor, see “Naming Generations and Levels” in the *Essbase Administration Services Online Help*.

Creating UDAs

You can create your own user-defined attributes for members. A *user-defined attribute (UDA)* is a word or phrase about a member. For example, you might create a UDA called Debit. Use UDAs in the following places:

- Calculation scripts. After you define a UDA, you can query a member for its UDA in a calculation script. For example, you could multiply all members with the UDA Debit by -1 so that they display as either positive or negative (depending on how the data is currently stored). See Chapter 27, "Developing Calculation Scripts."
- Data loading. You can change the sign of the data as it is loaded into the database based on its UDA. See "Flipping Field Signs" on page 404.

If you want to perform a calculation, selectively retrieve data based on attribute values, or provide full crosstab, pivot, and drill-down support in the spreadsheet, create attribute dimensions instead of UDAs. See "Comparing Attributes and UDAs" on page 190.

Follow these rules when creating UDAs:

- You can define multiple UDAs per member.
- You cannot set the same UDA twice for one member.
- You can set the same UDA for different members.
- A UDA name can be the same as a member, alias, level, or generation name. When you name UDAs, follow the same naming rules as for members. See "Understanding the Rules for Naming Dimensions and Members" on page 143.
- You cannot create a UDA on shared members.
- You cannot create a UDA on members of attribute dimensions.
- A UDA applies to the specified member only. Descendants and ancestors of the member do not automatically receive the same UDA.

➤ To add UDAs to a member, see "Working with UDAs" in the *Essbase Administration Services Online Help*.

Adding Comments

You can add comments to dimensions and members. A comment can be up to 255 characters long. Outline Editor displays comments to the right of the dimension or member in the following format:

```
/* comment */
```

➤ To add comments to a dimension or member, see “Setting Comments on Dimensions and Members” in the *Essbase Administration Services Online Help*.

Chapter 10

Working with Attributes

Attributes describe characteristics of data such as the size and color of products. Through attributes you can group and analyze members of dimensions based on their characteristics. This chapter describes how to create and manage attributes in an Analytic Server outline.

Note: The information in this chapter is designed for block storage databases. Some of the information is not relevant to aggregate storage databases. For detailed information on the differences between aggregate and block storage, see Chapter 57, "Comparison of Aggregate and Block Storage." For information on creating aggregate storage applications, see Chapter 58, "Aggregate Storage Applications, Databases, and Outlines."

This chapter contains the following topics:

You can find other information about attributes in relevant sections of this book.

Information Needed	More Information
Defining attributes through dimension build	"Building Attribute Dimensions and Associating Attributes" on page 434
Using attributes in partitions	• Chapter 13, "Designing Partitioned Applications" • Chapter 14, "Creating and Maintaining Partitions"
Using attributes in report writer	Chapter 32, "Developing Report Scripts"

Process for Creating Attributes

When working with attributes in Outline Editor perform the following tasks:

1. Create a new dimension. See "Adding Dimensions and Members to an Outline" on page 143. In the outline, position the attribute dimensions after all standard dimensions.
2. Tag the dimension as an attribute dimension and set attribute dimension type as text, numeric, Boolean, or date. See "Creating Attribute Dimensions" on page 160.
3. Add members to the attribute dimension. See "Adding Dimensions and Members to an Outline" on page 143.
4. Associate a base dimension with the attribute dimension. See "Understanding the Rules for Attribute Dimension Association" on page 184.
5. Associate members of the base dimension with members of the attribute dimension. See "Understanding the Rules for Attribute Member Association" on page 185.
6. If necessary, set up the attribute calculations. See "Calculating Attribute Data" on page 200.

Understanding Attributes

You can use the Analytic Services attribute feature to retrieve and analyze data not only from the perspective of dimensions, but also in terms of characteristics, or attributes, of those dimensions. For example, you can analyze product profitability

based on size or packaging, and you can make more effective conclusions by incorporating into the analysis market attributes such as the population size of each market region.

Such an analysis could tell you that decaffeinated drinks sold in cans in small (less than 6,000,000-population) markets are less profitable than you anticipated. For more details, you can filter the analysis by specific attribute criteria, including minimum or maximum sales and profits of different products in similar market segments.

Here are a few ways analysis by attribute provides depth and perspective, supporting better-informed decisions:

- You can select, aggregate, and report on data based on common features (attributes).
- By defining attributes as having a text, numeric, Boolean, or date type, you can filter (select) data using type-related functions such as AND, OR, and NOT operators and <, >, and = comparisons.
- You can use the numeric attribute type to group statistical values by attribute ranges; for example, population groupings such as <500,000, 500,000–1,000,000, and >1,000,000.
- Through the Attribute Calculations dimension automatically created by Analytic Services, you can view sums, counts, minimum or maximum values, and average values of attribute data. For example, when you enter Avg and Bottle into a spreadsheet, Analytic Services retrieves calculated values for average sales in bottles for all the column and row intersections on the sheet.
- You can perform calculations using numeric attribute values in calculation scripts and member formulas; for example, to determine profitability by ounce for products sized by the ounce.
- You can create crosstabs of attribute data for the same dimension, and you can pivot and drill down for detail data in spreadsheets.

  An attribute *crosstab* is a report or spreadsheet showing data consolidations across attributes of the same dimension. For example, the crosstab in Figure 59 displays product packaging as columns and the product size in ounces as rows. At their intersections, you see the profit for each combination of package type and size.

From this information, you can see which size-packaging combinations were most profitable in the Florida market.

Figure 59: Crosstab Example

Product Year Florida Profit Actual

Bottle	Can	Pkg Type	
32	946	N/A	946
20	791	N/A	791
16	714	N/A	714
12	241	2,383	2,624
Ounces	2,692	2,383	5,075

Understanding Attribute Dimensions

In the Sample Basic database, products have attributes that are characteristics of the products. For example, products have an attribute that describes their packaging. In the outline, you see these characteristics as two dimensions, the Products dimension, and the Pkg Type attribute dimension that is associated with it. An *attribute* dimension has the word Attribute next to its name in the outline.

Figure 60 shows part of the Sample Basic outline featuring the Product dimension and three attribute dimensions, Caffeinated, Ounces, and Pkg Type.

Figure 60: Outline Showing Base and Attribute Dimensions

```
Product {Caffeinated, Ounces, Pkg Type }
      100 (+)
            100-10 (+) {Caffeinated:True, Ounces:12, Pkg Type:Can }
            100-20 (+) {Caffeinated:True, Ounces:12, Pkg Type:Can }
            100-30 (+) {Caffeinated:False, Ounces:16, Pkg Type:Bottle }
      200 (+)
      300 (+)
      400 (+)
      Diet (~)
Caffeinated Attribute
      True
      False
Ounces Attribute
      32
      20
      16
      12
Pkg Type Attribute
      Bottle
      Can
```

In the outline, to the right of the Product dimension, the terms Caffeinated, Ounces, and Pkg Type show that these attribute dimensions are associated with the Product dimension.

A *standard* dimension is any dimension that is not an attribute dimension. When an attribute dimension is associated with a standard dimension, the standard dimension is the *base* dimension for that attribute dimension. In the outline in Figure 60, the Product dimension is the base dimension for the Caffeinated, Ounces, and Pkg Type attribute dimensions.

Note: Attribute dimensions and members are Dynamic Calc, so Analytic Services calculates attribute information at retrieval time. Attribute data is not stored in the database.

Understanding Members of Attribute Dimensions

Members of an attribute dimension are potential attributes of the members of the associated base dimension. After you associate a base dimension with an attribute dimension, you associate members of the base dimension with members of the associated attribute dimension. The Market dimension member Connecticut is associated with the 6000000 member of the Population attribute dimension. That makes 6000000 an attribute of Connecticut.

In the outline, the information next to a base dimension member shows the attributes of that member. In Figure 60, next to product "100-10, Caffeinated:True, Ounces:12, Pkg Type:Can" shows that product 100-10 has three attributes—product 100-10 has caffeine, it is sold in 12-ounce containers, and the containers are cans.

Understanding the Rules for Base and Attribute Dimensions and Members

There are several important rules regarding members of attribute dimensions and their base dimensions.

- You can tag only sparse dimensions as attribute dimensions.
- Before you can save an outline to the server, each attribute dimension must be associated with a standard, sparse dimension as its base dimension.
- Attribute dimensions must be the last dimensions in the outline.

- Attribute dimensions have a type setting—text, numeric, Boolean, or date. Text is the default setting. Although assigned at the dimension level, the type applies only to the level 0 members of the dimension. For more information, see "Understanding Attribute Types" on page 187.
- If you remove the attribute tag from a dimension, Analytic Services removes prefixes or suffixes from its member names. Prefixes and suffixes are not visible in the outline. For more information, see "Setting Prefix and Suffix Formats for Member Names of Attribute Dimensions" on page 196.
- A base dimension member can have many attributes, but only one attribute from each particular attribute dimension.

  For example, product 100-10 can have size and packaging attributes, but only one size and only one type of packaging.
- Analytic Services does not support attributes for Hybrid Analysis-enabled members.

You can use attribute values in calculations in the following comparisons:

- > (greater than)
- >= (greater than or equal to)
- < (less than)
- <= (less than or equal to)
- = = (equal to)
- <> or != (not equal to)
- IN

Understanding the Rules for Attribute Dimension Association

When you associate an attribute dimension with a standard dimension, the standard dimension is known as the *base* dimension for that attribute dimension.

- An attribute dimension must be associated with a sparse standard dimension.
- A standard dimension can be a base dimension for more than one attribute dimension.

- An attribute dimension can be associated with only one base dimension.

  For example, you might have a Size attribute dimension with members Small, Medium, and Large. If you associate the Size attribute dimension with the Product dimension, you cannot also associate the Size attribute dimension with the Market dimension. If you also want to track size-related information for the Market dimension, you must create another attribute dimension with a different name, for example, MarketSize, and associate the MarketSize attribute dimension with the Market dimension.

Understanding the Rules for Attribute Member Association

When you associate a member of an attribute dimension with a member of a base dimension, follow these rules:

- You cannot associate multiple members from the same attribute dimension with the same base dimension member. For example, the Bottle and Can package types cannot both be associated with the product 100-30.
- You can associate members from different attribute dimensions with the same member of a base dimension. For example, a decaffeinated cola product (100-30) sold in 16 ounce bottles has three attributes—Caffeinated:False; Ounces:16; and Pkg Type:Bottle.
- After attributes are associated with base dimension members, if you cut or copy and paste base dimension members to another location in the outline, the attribute associations are lost.
- Analytic Services does not require that each member of a base dimension be associated with a member of an attribute dimension.

- All base dimension members associated with members of a particular attribute dimension must be at the same level.

  For example, in Figure 61, all Market dimension members that have Population attributes are at level 0. You cannot associate East, which is a level 1 member, with a Population attribute since the other members of the Market dimension that have Population attributes are level 0 members.

  Figure 61: Association of Attributes with the Same Level Members of the Market Dimension

  ```
  Market {Population }
      East
      West
          California {Population:33000000 }
          Oregon {Population:6000000 }
          Washington {Population:6000000 }
          Utah {Population:3000000 }
          Nevada {Population:3000000 }
      South
      Central
  Population Attribute (Type: Numeric)
      Small
          3000000 (Alias: LT/= 3,000,000)
          6000000 (Alias: 3,000,001--6,000,000)
  ```

- The level 0 members of attribute dimensions are the only members that you can associate with base dimension members.

  For example, in the Population attribute dimension, you can associate only level 0 members such as 3000000, 6000000, and 9000000, with members of the Market dimension. You cannot associate a level 1 member such as Small.

  The name of the level 0 member of an attribute dimension is the attribute value. The only members of attribute dimensions that have attribute values are level 0 members.

  You can use the higher-level members of attribute dimensions to select and group data. For example, you can use Small, the level 1 member of the Population attribute dimension, to retrieve sales in both the 3000000 and 6000000 population categories.

Understanding Attribute Types

Attribute dimensions have a text, numeric, Boolean, or date type that enables different functions for grouping, selecting, or calculating data. Although assigned at the dimension level, the attribute type applies only to level 0 members of the attribute dimension.

- The default attribute type is text. Text attributes enable the basic attribute member selection and attribute comparisons in calculations. When you perform such comparisons, Analytic Services compares characters. For example, the package type Bottle is less than the package type Can because B precedes C in the alphabet. In Sample Basic, Pkg Type is an example of a text attribute dimension.
- The names of level 0 members of *numeric* attribute dimensions are numeric values. You can include the names (values) of numeric attribute dimension members in calculations. For example, you can use the number of ounces specified in the Ounces attribute to calculate profit per ounce for each product.

  You can also associate numeric attributes with ranges of base dimension values; for example, to analyze product sales by market population groupings—states with 3,000,000 population or less in one group, states with a population between 3,000,001 and 6 million in another group, and so on. See "Setting Up Member Names Representing Ranges of Values" on page 198.
- All Boolean attribute dimensions in a database contain only two members. The member names must match the settings for the database; for example, True and False. If you have more than one Boolean attribute dimension, you must specify a prefix or suffix member name format to ensure unique member names; for example, Caffeinated_True and Caffeinated_False. For a discussion of how to change Boolean names, see "Setting Boolean Attribute Member Names" on page 197.
- You can use date attributes to specify the date format—month-day-year or day-month-year—and to sequence information accordingly. For a discussion of how to change date formats, see "Changing the Member Names in Date Attribute Dimensions" on page 198. You can use date attributes in calculations. For example, you can compare dates in a calculation that selects product sales from markets established since 10-12-1999.

  Analytic Services supports date attributes from January 1, 1970 through January 1, 2038.

Comparing Attribute and Standard Dimensions

In general, attribute dimensions and their members are similar to standard dimensions and members. You can provide aliases and member comments for attributes. Attribute dimensions can include hierarchies and you can name generations and levels. You can perform the same spreadsheet operations on attribute dimensions and members as you can on standard dimensions and members; for example, to analyze data from different perspectives, you can retrieve, pivot, and drill down in the spreadsheet.

Table 12 describes major differences between attribute and standard dimensions and their members.

Table 12: Differences Between Attribute and Standard Dimensions

	Attribute Dimensions	**Standard Dimensions**
Storage	Must be sparse. Their base dimensions must also be sparse.	Can be dense or sparse
Storage property	Dynamic Calc only, therefore not stored in the database. The outline does not display this property.	Can be Store Data, Dynamic Calc and Store, Dynamic Calc, Never Share, or Label Only
Position in outline	Must be the last dimensions in the outline	Must be ahead of all attribute dimensions in the outline
Partitions	Cannot be defined along attribute dimensions, but you can use attributes to define a partition on a base dimension.	Can be defined along standard dimensions.
Formulas (on members)	Cannot be associated	Can be associated
Shared members	Not allowed	Allowed
Two-pass calculation member property	Not available	Available

Table 12: Differences Between Attribute and Standard Dimensions (Continued)

	Attribute Dimensions	Standard Dimensions
Two-pass calculation with run-time formula	If a member formula contains a run-time dependent function associated with an attribute member name, and the member with the formula is tagged as two-pass, calculation skips the member and issues a warning message. Run-time dependent functions include the following: @CURRMBR, @PARENT, @PARENTVAL, @SPARENTVAL, @MDPARENTVAL, @ANCEST, @ANCESTVAL, @SANCESTVAL, and @MDANCESTVAL.	Calculation is performed on standard members with run-time formulas and tagged two-pass.
Two-pass, multiple dimensions: Calculation order	Order of calculation of members tagged two-pass depends on order in outline. The last dimension is calculated last.	Calculation result is not dependent on outline order for members tagged two-pass in more than one dimension.
Two-pass calculation with no member formula	Calculation skipped, warning message issued. Thus member intersection of two-pass tagged members and upper level members may return different results from calculation on standard dimensions.	Available
Dense dynamic calc members in non-existing stored blocks	Calculation skips dense dimensions if they are on any non-existing stored block. To identify non-existing stored blocks, export the database or run query to find out whether block has any data.	Available
UDAs on members	Not allowed	Allowed
Consolidations	For all members, calculated through the Attribute Calculations dimension members: Sum, Count, Min, Max, and Avg.	Consolidation operation indicated by assigning the desired consolidation symbol to each member
Member selection facilitated by Level 0 member typing	Available types include text, numeric, Boolean, and date.	All members treated as text.

Table 12: Differences Between Attribute and Standard Dimensions (Continued)

	Attribute Dimensions	Standard Dimensions
Associations	Must be associated with a base dimension	N/A
Spreadsheet drill-downs	List the base dimension data associated with the selected attribute. For example, drilling down on the attribute Glass displays sales for each product packaged in glass, where Product is the base dimension for the Pkg Type attribute dimension.	List lower or sibling levels of detail in the standard dimensions. For example, drilling down on QTR1 displays a list of products and their sales for that quarter.

Comparing Attributes and UDAs

Attributes and UDAs both enable analysis based on characteristics of the data. Attributes provide much more capability than UDAs. Table 13 compares them. Checkmarks indicate the feature supports the corresponding capability.

Table 13: Comparing Attributes and UDAs

Capability	Attributes Feature	UDAs Feature
Data Storage		
You can associate with sparse dimensions.	✓	✓
You can associate with dense dimensions.		✓
Data Retrieval		
You can group and retrieve consolidated totals by attribute or UDA value. For example, associate the value High Focus Item to various members of the Product dimension and use that term to retrieve totals and details for just those members.	✓ Simple	✓ More difficult to implement, requiring additional calculation scripts or commands

Table 13: Comparing Attributes and UDAs (Continued)

Capability	Attributes Feature	UDAs Feature
You can categorize attributes in a hierarchy and retrieve consolidated totals by higher levels in the attribute hierarchy; for example, if each product has a specific size attribute such as 8, 12, 16, or 32, and the sizes are categorized as small, medium, and large. You can view the total sales of small products.	✓	✓ More difficult to implement
You can create crosstab views displaying aggregate totals of attributes associated with the same base dimension.	✓ You can show a crosstab of all values of each attribute dimension.	✓ You can only retrieve totals based on specific UDA values.
You can use Boolean operators AND, OR, and NOT with attribute and UDA values to further refine a query. For example, you can select decaffeinated drinks from the 100 product group.	✓	✓
Because attributes have a text, Boolean, date, or numeric type, you can use appropriate operators and functions to work with and display attribute data. For example, you can view sales totals of all products introduced after a specific date.	✓	
You can group numeric attributes into ranges of values and let the dimension building process automatically associate the base member with the appropriate range. For example, you can group sales in various regions based on ranges of their populations—less than 3 million, between 3 and 6 million, and so on.	✓	
Through the Attribute Calculations dimension, you can view aggregations of attribute values as sums, counts, minimums, maximums, and averages.	✓	
You can use an attribute in a calculation that defines a member. For example, you can use the weight of a product in ounces to define the profit per ounce member of the Measures dimension.	✓	

Table 13: Comparing Attributes and UDAs (Continued)

Capability	Attributes Feature	UDAs Feature
You can retrieve specific base members using attribute-related information.	✓ Powerful conditional and value-based selections	✓ Limited to text string matches only
Data Conversion		
Based on the value of a UDA, you can change the sign of the data as it is loaded into the database. For example, you can reverse the sign of all members with the UDA Debit.		✓
Calculation Scripts		
You can perform calculations on a member if its attribute or UDA value matches a specific value. For example, you can increase the price by 10% of all products with the attribute or UDA of Bottle.	✓	✓
You can perform calculations on base members whose attribute value satisfies conditions that you specify. For example, you can calculate the Profit per Ounce of each base member.	✓	

Designing Attribute Dimensions

Analytic Services provides more than one way to design attribute information into a database. Most often, defining characteristics of the data through attribute dimensions and their members is the best approach. The following sections discuss when to use attribute dimensions, when to use other features, and how to optimize performance when using attributes.

- "Using Attribute Dimensions" on page 193
- "Using Alternative Design Approaches" on page 193
- "Optimizing Outline Performance" on page 195

Using Attribute Dimensions

For the most flexibility and functionality, use attribute dimensions to define attribute data. Using attribute dimensions provides the following features:

- Sophisticated, flexible data retrieval

  You can view attribute data only when you want to, you can create meaningful summaries through crosstabs, and using type-based comparisons, you can selectively view just the data you want to see.

- Additional calculation functionality

  Not only can you perform calculations on the names of members of attribute dimensions to define members of standard dimensions, you can also access five different types of consolidations of attribute data—sums, counts, averages, minimums, and maximums.

- Economy and simplicity

  Because attribute dimensions are sparse, Dynamic Calc, they are not stored as data. Compared to using shared members, outlines using attribute dimensions contain fewer members and are easier to read.

For more information about attribute features, see "Understanding Attributes" on page 180.

Using Alternative Design Approaches

In some situations, consider one of the following approaches:

- UDAs. Although UDAs provide less flexibility than attributes, you can use them to group and retrieve data based on its characteristics. See "Comparing Attributes and UDAs" on page 190.
- Shared members. For example, to include a seasonal analysis in the Year dimension, repeat the months as shared members under the appropriate season; Winter: Jan (shared member), Feb (shared member), and so on. A major disadvantage of using shared members is that the outline becomes very large if the categories repeat a lot of members.
- Standard dimensions and members. Additional standard dimensions provide flexibility, but they add storage requirements and complexity to a database. For guidelines on evaluating the impact of additional dimensions, see "Analyzing and Planning" on page 80.

Table 14 describes situations where you might consider one of these alternative approaches for managing attribute data in a database.

Table 14: Considering Alternatives to Attribute Dimensions

Situation	Alternative to Consider
Analyze attributes of dense dimensions	UDAs or shared members.
Perform batch calculation of data	Shared members or members of separate, standard dimensions.
Define the name of a member of an attribute dimension as a value as that results from a formula	Shared members or members of separate, standard dimensions
Define attributes that vary over time	Members of separate, standard dimensions. For example, to track product maintenance costs over a period of time, the age of the product at the time of maintenance is important. However, using the attribute feature you could associate only one age with the product. You need multiple members in a separate dimension for each time period that you want to track.
Minimize retrieval time with large numbers of base-dimension members	Batch calculation with shared members or members of separate, standard dimensions.

Optimizing Outline Performance

Outline layout and content can affect attribute calculation and query performance. For general outline design guidelines, see "Designing an Outline to Optimize Performance" on page 100.

To optimize attribute query performance, consider the following design tips:

- Ensure that attribute dimensions are the only sparse Dynamic Calc dimensions in the outline.
- Locate sparse dimensions after dense dimensions in the outline. Place the most-queried dimensions at the beginning of the sparse dimensions and attribute dimensions at the end of the outline. In most situations, the base dimensions are the most queried dimensions.

For information on optimizing calculation of outlines containing attributes, see "Optimizing Calculation and Retrieval Performance" on page 205.

Building Attribute Dimensions

To build an attribute dimension, first tag the dimension as attribute and assign the dimension a type. Then associate the attribute dimension with a base dimension. Finally, associate each level 0 member of the attribute dimension with a member of the associated base dimension.

➤ To build an attribute dimension, see "Defining Attributes" in the *Essbase Administration Services Online Help*.

➤ To view the dimension, attribute value and attribute type of a specific attribute member, use any of the following methods:

Tool	Topic	Location
Administration Services	Viewing Attribute Information in Outlines	*Essbase Administration Services Online Help*
MaxL	**query database**	*Technical Reference*
ESSCMD	GETATTRINFO	*Technical Reference*

Setting Member Names in Attribute Dimensions

All member names in an outline must be unique. When you use the attribute feature, Analytic Services establishes some default member names. These default names might duplicate names that already exist in the outline. You can change these system-defined names for the database and can establish other settings for members of attribute dimensions in the database. The outline does not show the full attribute names. You can see and use the full attribute names anywhere you select members, such as when you define partitions or select information to be retrieved.

Define the member name settings before you define or build the attribute dimensions. Changing the settings after the attribute dimensions and members are defined could result in invalid member names.

The following sections describe how to work with the names of members of attribute dimensions:

- "Setting Prefix and Suffix Formats for Member Names of Attribute Dimensions" on page 196
- "Setting Boolean Attribute Member Names" on page 197
- "Changing the Member Names in Date Attribute Dimensions" on page 198
- "Setting Up Member Names Representing Ranges of Values" on page 198
- "Changing the Member Names of the Attribute Calculations Dimension" on page 200

Note: If you partition on outlines containing attribute dimensions, the name format settings of members described in this section must be identical in the source and target outlines.

Setting Prefix and Suffix Formats for Member Names of Attribute Dimensions

The names of members of Boolean, date, and numeric attribute dimensions are values. It is possible to encounter duplicate attribute values in different attribute dimensions.

- Boolean example. If you have more than one Boolean attribute dimension in an outline, the two members of each of those dimensions have the same names, by default, True and False.

- Date example. If you have more than one date attribute dimension, some member names in both dimensions could be the same. For example, the date that a store opens in a certain market could be the same as the date a product was introduced.
- Numeric example. 12 can be the attribute value for the size of a product and 12 could also be the value for the number of packing units for a product. This example results in two members with the same name—12.

Because Analytic Services does not allow duplicate member names, you can define unique names by attaching a prefix or suffix to member names in Boolean, date, and numeric attribute dimensions in the outline. For example, by setting member names of attribute dimensions to include the dimension name as the suffix, attached by an underscore, the member value 12 in the Ounces attribute dimension assumes the unique, full attribute member name, 12_Ounces.

By default, Analytic Services assumes that no prefix or suffix is attached to the names of members of attribute dimensions.

The convention that you select applies to the level 0 member names of all numeric, Boolean, and date attribute dimensions in the outline. You can define aliases for these names if you wish to display shorter names in retrievals.

➤ To define prefix and suffix formats, see "Defining a Prefix or Suffix Format for Members of Attribute Dimensions" in the *Essbase Administration Services Online Help*.

Setting Boolean Attribute Member Names

When you set the dimension type of an attribute dimension as Boolean, Analytic Services automatically creates two level 0 members with the names specified for the Boolean attribute settings. The initial Boolean member names in a database are set as True and False. If you want to change these default names, for example, to Yes and No, you must define the member names for Boolean attribute dimensions before you create any Boolean attribute dimensions in the database.

Before you can set an attribute dimension type as Boolean, you must delete all existing members in the dimension.

➤ To define the database setting for the names of members of Boolean attribute dimensions, see "Setting Member Names for Boolean Attribute Dimensions" in the *Essbase Administration Services Online Help*.

Changing the Member Names in Date Attribute Dimensions

You can change the format of members of date attribute dimensions. For example, you can use the following date formats:

- mm-dd-yyyy displays the month before the day; for example, October 18, 1999 is displayed as 10-18-1999.
- dd-mm-yyyy displays the day before the month; for example, October 18, 1999 is displayed as 18-10-1999.

If you change the date member name format, the names of existing members of date attribute dimensions may be invalid. For example, if the 10-18-1999 member exists and you change the format to dd-mm-yyyy, outline verification will find this member invalid. If you change the date format, you must rebuild the date attribute dimensions.

➤ To change the names of the members in date attribute dimensions, see "Setting the Member Name Format of Date Attribute Dimensions" in the *Essbase Administration Services Online Help*.

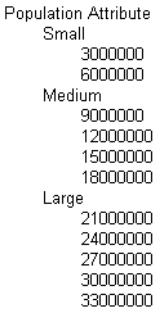
Setting Up Member Names Representing Ranges of Values

Members of numeric attribute dimensions can represent single numeric values or ranges of values:

- Single value example: the member 12 in the Ounces attribute dimension represents the single numeric value 12; you associate this attribute with all 12-ounce products. The outline includes a separate member for each size; for example, 16, 20, and 32.

- Range of values example: the Population attribute dimension:

  Figure 62: Population Attribute Dimension and Members

  In this outline, the members of the Population attribute dimension represent ranges of population values in the associated Market dimension. The 3000000 member represents populations from zero through 3,000,000; the 6000000 member represents populations from 3,000,001 through 6,000,000; and so on. Each range includes values greater than the name of the preceding member up to and including the member value itself. A setting for the outline establishes that each numeric member represents the top of its range.

  You can also define this outline setting so that members of numeric attribute dimensions are the bottoms of the ranges that they represent. For example, if numeric members are set to define the bottoms of the ranges, the 3000000 member represents populations from 3,000,000 through 5,999,999 and the 6000000 member represents populations from 6,000,000 through 8,999,999.

When you build the base dimension, Analytic Services automatically associates members of the base dimension with the appropriate attribute range. For example, if numeric members represent the tops of ranges, Analytic Services automatically associates the Connecticut market, with a population of 3,269,858, with the 6000000 member of the Population attribute dimension.

In the dimension build rules file, specify the size of the range for each member of the numeric attribute dimension. In the above example, each attribute represents a range of 3,000,000.

➤ To set up ranges in numeric attribute dimensions, see “Assigning Member Names to Ranges of Values” in the *Essbase Administration Services Online Help*.

Changing the Member Names of the Attribute Calculations Dimension

To avoid duplicating names in an outline, you may need to change the name of the Attribute Calculations dimension or its members. For more information about this dimension, see "Understanding the Attribute Calculations Dimension" on page 201.

Regardless of the name that you use for a member, its function remains the same. For example, the second (Count) member always counts, no matter what you name it.

➤ To change the names of the members in the Attribute Calculations dimension, see "Changing Member Names of Attribute Calculations Dimensions" in the *Essbase Administration Services Online Help*.

Calculating Attribute Data

Analytic Services calculates attribute data dynamically at retrieval time, using members from a system-defined dimension created specifically by Analytic Services. Using this dimension, you can apply different calculation functions, such as a sum or an average, to the same attribute. You can also perform specific calculations on members of attribute dimensions; for example, to determine profitability by ounce for products sized by the ounce.

The following information assumes that you understand the concepts of attribute dimensions and Analytic Services calculations, including dynamic calculations.

This section includes the following sections:

Understanding the Attribute Calculations Dimension

When you create the first attribute dimension in the outline, Analytic Services also creates the Attribute Calculations dimension comprising five members with the default names Sum, Count, Min (minimum), Max (maximum), and Avg (average). You can use these members in spreadsheets or in reports to dynamically calculate and report on attribute data, such as the average yearly sales of 12-ounce bottles of cola in the West.

The Attribute Calculations dimension is not visible in the outline. You can see it wherever you select dimension members, such as in the Spreadsheet Add-in.

The attribute calculation dimension has the following properties:

- System-defined. When you create the first attribute dimension in an application, Analytic Services creates the Attribute Calculations dimension and its members (Sum, Count, Min, Max, and Avg). Each member represents a type of calculation to be performed for attributes. For a discussion of calculation types, see "Understanding the Default Attribute Calculations Members" on page 203.
- Label only. Like all label only dimensions, the Attribute Calculations dimension shares the value of its first child, Sum. For more information on the label only dimension property, see "Member Storage Properties" on page 99.
- Dynamic Calc. The data in the Attribute Calculations dimension is calculated when a user requests it and is then discarded. You cannot store calculated attribute data in a database. For a comprehensive discussion on dynamic calculations, see Chapter 25, "Dynamically Calculating Data Values".
- Not displayed in Outline Editor. The Attribute Calculations dimension is not displayed in Outline Editor. Members from this dimension can be viewed in spreadsheets and in reports.

There is no consolidation along attribute dimensions. You cannot tag members from attribute dimensions with consolidation symbols (for example, + or -) or with member formulas in order to calculate attribute data. As Dynamic Calc members,

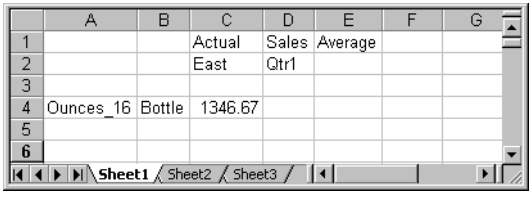
attribute calculations do not affect the batch calculation in terms of time or calculation order. To calculate attribute data at retrieval time, Analytic Services performs the following tasks:

1. Finds the base-dimension members that are associated with the specified attribute-dimension members present in the current query
2. Dynamically calculates the sum, count, minimum, maximum, or average for the attribute-member combination for the current query
3. Displays the results in the spreadsheet or report
4. Discards the calculated values—that is, the values are not stored in the database

Note: Analytic Services excludes #MISSING values when calculating attribute data.

For example, as shown in Figure 63, a spreadsheet user specifies two members of attribute dimensions (Ounces_16 and Bottle) and an Attribute Calculations member (Avg) in a spreadsheet report. Upon retrieval, Analytic Services dynamically calculates the average sales values of all products associated with these attributes for the current member combination (Actual -> Sales -> East -> Qtr1):

Figure 63: Retrieving an Attribute Calculations Member



For information on accessing calculated attribute data, see "Accessing Attribute Calculations Members Using the Spreadsheet" on page 205.

Understanding the Default Attribute Calculations Members

The Attribute Calculations dimension contains five members used to calculate and report attribute data. These members are as follows:

- Sum calculates a sum, or total, of the values for a member with an attribute or combination of attributes. Supports non-additive consolidations such as multiplication, and two-pass measures.

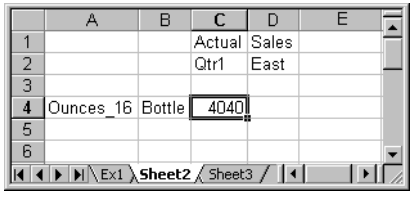
  Note: The Sum member totals members based on their consolidation property or formula. For example, the Sum member uses the following formula to consolidate the profit percentages of 12-ounce products:

$$\text{Sum of Profit\% of 12-ounce products} = \frac{\text{Sum of Profit of base members with the Ounces attribute 12}}{\text{Sum of Sales of base members with the Ounces attribute 12}} * 100$$

  This calculation is not the sum of all percentages for all base-dimension members with the Ounces attribute 12.

  The default calculation for attributes is Sum. If a spreadsheet user specifies a member of an attribute dimension in a spreadsheet but does not specify a member of the Attribute Calculations dimension, Analytic Services retrieves the sum for the specified attribute or combination of attributes. For example, in the spreadsheet view in Figure 64, the value in cell C4 represents the sum of sales values for the attributes Ounces_16 and Bottle for the current member combination (Actual -> Sales -> East -> Qtr1), even though the Sum member is not displayed in the sheet.

  Figure 64: Retrieving the Default Attribute Calculations Member



- Count calculates the number of members with the specified attribute or combination of attributes, for which a data value exists. Count includes only those members that have data blocks in existence. To calculate a count of all

members with certain attributes, regardless of whether or not they have data values, use the @COUNT function in combination with the @ATTRIBUTE function. For more information, see the *Technical Reference*.

- Avg calculates a mathematical mean, or average, of the non-missing values for an specified attribute or combination of attributes (Sum divided by Count).
- Min calculates the minimum data value for a specified attribute or combination of attributes.
- Max calculates the maximum data value for a specified attribute or combination of attributes.

Note: Each of these calculations excludes #MISSING values.

You can change these default member names, subject to the same naming conventions as standard members. For a discussion of Attribute Calculations member names, see "Changing the Member Names of the Attribute Calculations Dimension" on page 200.

Viewing an Attribute Calculation Example

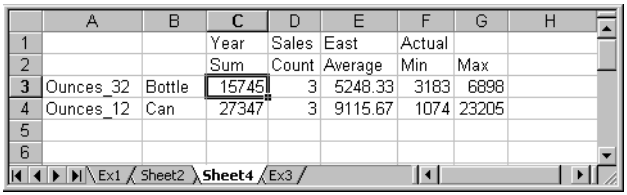
As an example of how Analytic Services calculates attribute data, consider the following yearly sales data for the East:

Table 15: Sample Attribute Data

Base-Dimension Member	Associated Attributes	Sales Value for Attribute-Member Combination
Cola	Ounces_12, Can	23205
Diet Cola	Ounces_12, Can	3068
Diet Cream	Ounces_12, Can	1074
Grape	Ounces_32, Bottle	6398
Orange	Ounces_32, Bottle	3183
Strawberry	Ounces_32, Bottle	5664

A spreadsheet report showing calculated attribute data might look like the following illustration:

Figure 65: Sample Spreadsheet with Attribute Data



	A	B	C	D	E	F	G	H
1			Year	Sales	East	Actual		
2			Sum	Count	Average	Min	Max	
3	Ounces_32	Bottle	15745	3	5248.33	3183	6898	
4	Ounces_12	Can	27347	3	9115.67	1074	23205	
5								
6								

Ex1 | Sheet2 | Sheet4 | Ex3

As shown in the figure above, you can retrieve multiple Attribute Calculations members for attributes. For example, you can calculate Sum, Count, Avg, Min, and Max for 32-ounce bottles and cans.

Accessing Attribute Calculations Members Using the Spreadsheet

You can access members from the Attribute Calculations dimension in Spreadsheet Add-in. From the spreadsheet, users can view Attribute Calculations dimension members using any of the following methods:

- Entering members directly into a sheet
- Selecting members from the Query Designer
- Entering members as an EssCell parameter

For more information on accessing calculated attribute data from the spreadsheet, see the *Essbase Spreadsheet Add-in User's Guide*.

Optimizing Calculation and Retrieval Performance

To optimize attribute calculation and retrieval performance, consider the following considerations:

- The calculation order for attribute calculations is the same as the order for dynamic calculations. For an outline of calculation order, see "Calculation Order for Dynamic Calculation" on page 553.
- Since Analytic Services calculates attribute data dynamically at retrieval time, attribute calculations do not affect the performance of the overall (batch) database calculation.

- Tagging base-dimension members as Dynamic Calc may increase retrieval time.
- When a query includes the Sum member and an attribute-dimension member whose associated base-member is tagged as two-pass, retrieval time may be slow.
- To maximize attribute retrieval performance, use any of the following techniques:
  - Configure the outline using the tips in "Optimizing Outline Performance" on page 195.
  - Drill down to the lowest level of base dimensions before retrieving data. For example, in Spreadsheet Add-in, turn on the Navigate Without Data feature, drill down to the lowest level of the base dimensions included in the report, and then retrieve data.
  - When the members of a base dimension are associated with several attribute dimensions, consider grouping the members of the base dimension according to their attributes. For example, in the Sample Basic database, you could group all 8-ounce products. Grouping members by attribute may decrease retrieval time.

Using Attributes in Calculation Formulas

In addition to using the Attribute Calculations dimension to calculate attribute data, you can also use calculation formulas on members of standard or base dimensions to perform specific calculations on members of attribute dimensions; for example, to determine profitability by ounce for products sized by the ounce.

You cannot associate formulas with members of attribute dimensions.

Note: Some restrictions apply when using attributes in formulas associated with two-pass members. For details, see the rows about two-pass calculations in Table 12 on page 188.

You can use the following functions to perform specific calculations on attributes:

Type of Calculation	Function to Use
Generate a list of all base members with a specific attribute. For example, you can generate a list of members that have the Bottle attribute, and then increase the price for those members.	@ATTRIBUTE
Return the value of the level 0 attribute member that is associated with the base member being calculated. • From a numeric or date attribute dimension (using @ATTRIBUTEVAL) • From a Boolean attribute dimension (using @ATTRIBUTEBVAL) • From a text attribute dimension (using @ATTRIBUTESVAL) For example, you can return the numeric value of a size attribute (for example, 12 for the member 12 under Ounces) for the base member being calculated (for example, Cola).	@ATTRIBUTEVAL @ATTRIBUTEBVAL @ATTRIBUTESVAL
Convert a date string to numbers for a calculation. For example, you can use @TODATE in combination with the @ATTRIBUTEVAL function to increase overhead costs for stores opened after a certain date.	@TODATE
Generate a list of all base dimension members associated with attributes that satisfy the conditions that you specify. For example, you can generate a list of products that are greater than or equal to 20 ounces, and then increase the price for those products.	@WITHATTR

Note: For syntax information and examples for these functions, see the *Technical Reference*. For an additional example using @ATTRIBUTEVAL in a formula, see "Calculating an Attribute Formula" on page 514.

Understanding Attribute Calculation and Shared Members

Attribute calculations start at level 0 and stop at the first stored member. Therefore, if your outline has placed a real member in between two shared members in a an outline hierarchy, the calculation results may not include the higher shared member.

For example:

```
Member 1 (stored)
     Member A (stored)
          Member 2 (shared)
Member B (stored)
    Member 1 (shared member whose stored member is Member 1 above)
```

In this example, when an attribute calculation is performed, the calculation starts with level 0 Member 2, and stops when it encounters the first stored member, Member A. Therefore, Member 1 would not be included in the calculation.

Avoid mixing shared and stored members to avoid unexpected results with attribute calculation. For this example, if Member 2 were not shared, or Member 1 did not have a corresponding shared member elsewhere in the outline, calculation results would be as expected.

Chapter 11

Linking Objects to Analytic Services Data

This chapter describes how you can link various kinds of data with any cell in an Analytic Services database, using a linked reporting object (LRO). This ability is similar to the file attachment features in an e-mail software package.

An LRO provides improved support for planning and reporting applications and can enhance your data analysis capabilities.

Note: The information in this chapter is not relevant to aggregate storage databases. For detailed information on the differences between aggregate and block storage, see Chapter 57, "Comparison of Aggregate and Block Storage."

This chapter contains the following sections:

Understanding LROs

LROs are objects that you associate with specific data cells in an Analytic Services database. Users create linked objects through Spreadsheet Add-in and Spreadsheet Services by selecting a data cell and choosing a menu item. There is no limit to the number of objects you can link to a cell. The objects are stored on the Analytic Server where they are available to any user with the appropriate access permissions. Users retrieve and edit the objects through the Spreadsheet Add-in

and Spreadsheet Services Linked Objects Browser feature, enabling them to view objects linked to the selected cell. For the maximum sizes of the types of linked objects described in Table 16, see Appendix A, "Limits."

Table 16: Types of Linked Objects

Object Type	Description
Cell note	A text annotation
File	An external file, such as a Microsoft Word document, an Excel spreadsheet, a scanned image, an audio clip, or an HTML file (for example, `mypage.htm`).

URL	An acronym for Uniform Resource Locator. A string that identifies the location of a resource on the World Wide Web, such as a document, image, downloadable file, service, electronic mailbox, or other resource.
	For example:
	`http://www.hyperion.com`
	`ftp://ftp.hyperion.com`
	`file:///D|/ESSBASE/docs/index.htm`

Linked partition	A set of data cells that you can link to in another Analytic Services database.

For example, a sales manager may attach cell notes to recently updated budget items. A finance manager might link a spreadsheet containing supporting data for this quarter's results. A product manager might link bitmap images of new products. A sales manager may link the URL of a company's Web site to quickly access the information on the Web site.

Understanding LRO Types and Data Cells

LROs are linked to data cells—not to the data contained in the cells. The link is based on a specific member combination in the database. Adding or removing links to a cell does not affect the cell contents.

When a user links an object to a cell, Analytic Services stores in the object catalog information about the type of object, the name of the last user to modify the object, and the date the object was modified.

How Analytic Services stores the LRO depends on the LRO type:

- If the object is a cell note, the text is stored as part of the object description in the catalog entry.
- If the object is a file, the Analytic Services stores the contents of the file in the database directory on the Analytic Server, giving it a `.LRO` extension. Analytic Services imposes no restrictions on the data formats of linked files and performs no file-type checking. It is up to the user's client machine to render the file after retrieving it from the Analytic Server.
- If the object is a URL, Analytic Services stores the URL string as part of the object description in the catalog entry. Analytic Services does not check the syntax of the URL until the user tries to view it. At that time, Analytic Services does a preliminary syntax check; then the default Web browser checks for the existence of the URL.
- If the object is a linked partition, it is available through the Analytic Services Partitioning feature.

Before you perform any tasks related to LROs, be aware of these facts:

- Analytic Services uses the database index to locate and retrieve linked objects. If you clear all data values from a database, the index is deleted and so are the links to linked objects. If you restructure a database, the index is preserved and so are the links to linked objects.
- Shared members share data values but do not share LROs. This is because LROs are linked to specific member combinations and shared members do not have identical member combinations. If you want a given object to be linked to shared members, you must link it to each shared member individually.
- You cannot change the member combination associated with any linked object. To move an object to another member combination, first delete it, then use Spreadsheet Add-in or Spreadsheet Services to re-link the object to the desired member combination.

Setting Up Permissions for LROs

Users who add, edit, and delete LROs through client interfaces need to have the appropriate security permissions in the active database. If the object is a linked partition, the user must also have the required permissions in the database containing the linked partition.

Table 17 lists the permissions required for several different tasks.

Table 17: Permissions Required for LRO Tasks

Task	Permission
Add a linked object to a database	Read/Write
View an existing linked object	Read
Edit an existing linked object	Read/Write
Delete a linked object	Read/Write
Export the LRO catalog to a file	Read
Import the LROs from the LRO-catalog file	Read/Write

Sometimes you might want to prevent users from linking files to data cells without changing user access to other data in a database. You can accomplish this by setting the maximum file size for linked files to 1. Users can then create cell notes, link to a URL, or view linked partitions but can only attach very small files (under 1 kilobyte).

➤ To set the maximum LRO file size for an application, see "Limiting LRO File Sizes" in *Essbase Administration Services Online Help*.

Viewing and Deleting LROs

Users work with LROs on a cell-by-cell basis through Spreadsheet Add-in or Spreadsheet Services. That is, they select a cell and open the Linked Object Browser, which displays the objects linked to the selected cell. With Administration Services, you can view LROs and you can delete all LROs for the entire database. You can also view LROs based on selection criteria such as user name and last modification date. For example, you might want to purge all objects that are older than a certain date, or remove the objects belonging to a user who has left the company.

➤ To view a list of the linked objects for a database, use any of the following methods:

Tool	Topic	Location
Administration Services	Managing LROs	*Essbase Administration Services Online Help*
MaxL	**query database**	*Technical Reference*
ESSCMD	LISTLINKEDOBJECTS	*Technical Reference*

➤ To delete the linked objects for a database, use any of the following methods:

Tool	Topic	Location
Administration Services	Managing LROs	*Essbase Administration Services Online Help*
MaxL	**alter database**	*Technical Reference*
ESSCMD	PURGELINKEDOBJECTS	*Technical Reference*

Exporting and Importing LROs

To improve backup and data-migration capabilities, you can export and re-import LROs from data intersections in a database.

➤ To export and import linked objects for a database, use any of the following methods:

Tool	Topic	Location
Administration Services	Exporting LROs Importing LROs	*Essbase Administration Services Online Help*
MaxL	**export lro** **import lro**	*Technical Reference*

Limiting LRO File Sizes for Storage Conservation

Because Analytic Services stores linked files in a repository on the server, you might want to limit the size of files that users can link. By default, the size is unlimited. Limiting the size prevents a user from taking up too much of the server resources by storing extremely large objects. You can set the maximum linked file size for each application. If a user attempts to link a file that is larger than the limit, an error message displays.

To prevent users from attaching anything except very small files, enter 1. Setting the file size to 1, lets users link only cell notes, URLs, and files less than 1 kilobyte in size.

Note: The maximum file size setting applies only to linked files and does not affect cell notes or URLs. The lengths of the cell note, URL string, and LRO descriptions are fixed. For the maximum sizes of these objects, see Appendix A, "Limits."

➤ To limit the size of a linked object, use any of the following methods:

Tool	Topic	Location
Administration Services	Limiting LRO File Sizes	*Essbase Administration Services Online Help*
MaxL	**alter application**	*Technical Reference*
ESSCMD	SETAPPSTATE	*Technical Reference*

Chapter 12

Designing and Building Currency Conversion Applications

You use the Analytic Services currency conversion feature to translate financial data from one currency into another currency. Currency conversion facilitates comparisons among countries, and enables consolidation of financial data from locations that use different currencies. This feature can be licensed as an "add-on" to Analytic Server.

For example, consider an organization that analyzes profitability data from the UK, reported in pounds, and from Japan, reported in yen. Comparing *local currency* profitability figures side-by-side in a spreadsheet is meaningless. To understand the relative contribution of each country, you need to convert pounds into yen, yen into pounds, or both figures into another currency.

As another example, reporting total profitability for North America requires standardization of the local currency values that constitute the North America total. Assuming that the United States, Mexico, and Canada consolidate into Total North America, the profitability total is meaningless if data is kept in local currencies. The Total North America sum is meaningful only if local currencies are converted to a common currency prior to consolidation.

The Analytic Services installation includes the option to install the Sample currency application, which consists of two databases, Interntl and Xchgrate. If you do not have access to these databases, contact your Analytic Services administrator. For information about installing Sample currency applications, see the *Essbase Analytic Services Installation Guide*.

Note: The information in this chapter is not relevant to aggregate storage databases. For detailed information on the differences between aggregate and block storage, see Chapter 57, "Comparison of Aggregate and Block Storage."

This chapter contains the following topics:

- "Conversion Methods" on page 222
- "Building Currency Conversion Applications and Performing Conversions" on page 222

About the Sample Currency Application

The Sample currency application builds on the business scenario introduced in Chapter 5, "Case Study: Designing a Single-Server, Multidimensional Database," as the Beverage Company (TBC) expands its business outside the United States. TBC adds the following markets:

- Three locations in Canada: Toronto, Vancouver, and Montreal
- Four locations in Europe: the UK, Germany, Switzerland, and Sweden

In addition, TBC adds a new member, US, which is a consolidation of data from the United States regions: East, West, South, and Central.

Data for each TBC market location is captured in local currency. U.S. dollar values are derived by applying exchange rates to local values.

TBC needs to analyze actual data in two ways:

- Actuals are converted at actual exchange rates.
- Actuals are converted at budget exchange rates to analyze variances due to exchange rates.

After all actuals are processed, budget data is converted with budget exchange rates.

The TBC currency application consists of the main database (Interntl) and the currency database (Xchgrate). On Analytic Server, the databases are in the Sample application. If you do not have access to the databases, contact your Analytic Services administrator. For information about installing Sample currency applications, see the *Essbase Analytic Services Installation Guide*.

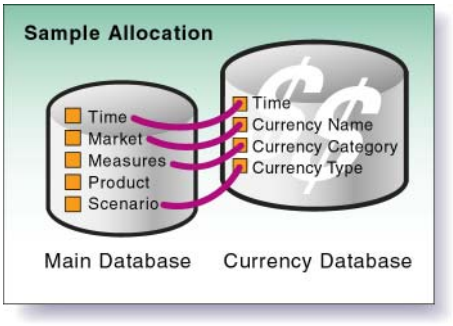
Structure of Currency Applications

In a business application requiring currency conversion, the *main database* is divided into at least two slices. One slice handles input of the local data, and another slice holds a copy of the input data converted to a common currency.

Analytic Services holds the exchange rates required for currency conversion in a separate *currency database*. The currency database outline, which is automatically generated by Analytic Services from the main database after you assign the necessary tags, typically maps a given conversion ratio onto a section of the main database. After the currency database is generated, it can be edited just like any other Analytic Services database.

The relationship between the main database and the currency database is illustrated in Figure 66.

Figure 66: Currency Application Databases



Main Database

To enable Analytic Services to generate the currency database outline automatically, you modify dimensions and members in the main database outline. In the Sample currency application, the main database is Interntl.

The main database outline can contain from 3 to *n* dimensions. At a minimum, the main database must contain the following dimensions:

- A dimension tagged as time. Tagging a dimension as time generates a dimension in the currency database that is identical to the time dimension in the main database. In the Sample Interntl database, the dimension tagged as time is Year.

- A dimension tagged as accounts. Tagging a dimension as accounts and assigning currency categories to its members creates a dimension in the currency database that contains members for each of the individual currency categories. Category assignment enables the application of different exchange rates to various accounts or measures. In the Sample Interntl database, the dimension tagged as accounts is Measures.

  Each descendant of a member inherits the currency category tag of its ancestor. A member or sub-branch of members can also have its own category.

  For example, profit and loss (P&L) accounts may use exchange rates that differ from the rates used with balance sheet accounts. In addition, some accounts may not require conversion. For example, in the Sample Interntl database, members such as Margin% and Profit% require no conversion. You tag members not to be converted as No Conversion. The No Conversion tag is not inherited.

- A market-related dimension tagged as country. Tagging a dimension as country and assigning currency names to individual countries creates a member in the currency database for each currency. In the Sample Interntl database, the Market dimension is tagged as country. The currency name for this dimension is USD (U.S. dollars), because all local currencies must be converted to USD, the company's common currency.

  Because multiple members can have the same currency name, the number of currency names is typically less than the total number of members in the dimension. As shown in Table 18, the Sample Interntl database uses only six currency names for the 15 members in the Market dimension. Each of the children of the member Europe use a different currency and, therefore, must be assigned an individual currency name. However, the US dimension and its four regional members all use the same currency. The same is true of the

Canada member and its three city members. When the children of a given member share a single currency, you need to define a currency name for only the parent member.

Table 18: Interntl Database Currency Names

Dimensions and Members	Currency Name
Market - Country	USD (U.S. dollar)
US	
East	
West	
South	
Central	
Canada	CND (Canadian dollar)
Toronto	
Vancouver	
Montreal	
Europe	
UK	GBP (British pound)
Germany	EUR (Euro)
Switzerland	CHF (Swiss franc)
Sweden	SEK (Swedish krona)

When preparing a main database outline for currency conversion, you can create an optional *currency partition* to tell Analytic Services which slice of the database holds local currency data and which slice of the database holds data to be converted. The dimension that you tag as currency partition contains members for both local currency values and converted values. Local currency data is converted to common currency data using currency conversion calculation scripts. In the Sample Interntl database, the Scenario dimension is the currency partition dimension.

For instructions on how to use currency partition dimensions, see "Keeping Local and Converted Values" on page 225.

Note: A currency conversion partition applies only to the currency conversion option. It is not related to the Partitioning option that enables data to be shared between databases by using a replicated, linked, or transparent partition.

The *Essbase Spreadsheet Add-in User's Guide* provides examples of ad hoc currency reporting capabilities. Report scripts enable the creation of reports that convert data when the report is displayed, as discussed under "Converting Currencies in Report Scripts" on page 228.

Note: For a list of methods used to create the main database outline, see "Creating Main Database Outlines" on page 223.

Currency Database

By assigning currency tags to members in the main database outline, you enable Analytic Services to generate the currency database automatically. In the Sample currency application, the currency database is Xchgrate.

A currency database always consists of the following three dimensions, with an optional fourth dimension:

- A dimension tagged as time, which is typically the same as the dimension tagged as time in the main database. This allows the currency database to track currency fluctuations over time and to accurately convert various time slices of the main database. In the Sample Xchgrate database, the dimension tagged as time is Year.

  Each member of the time dimension in the main database must be defined in the currency database. Values by time period in the main database are usually converted to the exchange rates of their respective time period from the currency database (although you can convert data values against the exchange rate of any period).

- A dimension tagged as country, which contains the names of currencies relevant to the markets (or countries) defined in the main database. Each currency name defined in the main database must also exist in the currency database. The currency names define the country-to-exchange rate mapping when conversion occurs.

In the Sample Xchgrate database, the country dimension is CurName. CurName contains the following currency names:

Table 19: Xchgrate Database Currency Names

Dimension and Members	Alias Name
CurName - Country	
USD	US dollar
CND	Canadian dollar
GBP	British pound
EUR	Euro
CHF	Swiss franc
SEK	Swedish krona

- A dimension tagged as accounts, which enables the application of various rates to members of the dimension tagged as accounts in the main database. The categories defined for the accounts dimension in the main database are used to form the members in the accounts dimension of the currency database. For example, it may be necessary to convert Gross Profit and Net Profit using one category of rates, while other accounts use a different set of rates.

  In the Sample Xchgrate database, the dimension tagged as accounts is CurCategory, and the account categories included are P&L (Profit & Loss) and B/S (Balance Sheet).

- A currency database typically includes an optional currency type dimension, which enables different scenarios for currency conversion. Typically, an application has different exchange rates for different scenarios, such as actual, budget, and forecast. To convert data between scenarios, simply select which type of rate to use.

  The currency type dimension is created when you generate the currency outline and is not directly mapped to the main database. Therefore, member names in this dimension are not required to match member names of the main database.

  In the Sample Xchgrate database, the currency type dimension is CurType. CurType includes actual and budget scenarios.

Note: For information about creating the currency database outline, see "Building Currency Conversion Applications and Performing Conversions" on page 222.

Conversion Methods

Different currency applications have different conversion requirements. Analytic Services supports two conversion methods:

- Overwriting local values with converted values.

  Some applications require only converted values to be stored in the main database. Local values are entered and the conversion operation overwrites local values with common currency values. This method assumes that there is no requirement for reporting or analyzing local currencies.

  Because this operation overwrites data, you must load local values and recalculate the data each time you perform a conversion. This method is useful only when you want to perform a single (not an ongoing) conversion.

- Keeping local and converted values.

  Most applications require data to be stored in both local and common currency (converted) values. This method permits reporting and analyzing local data. In addition, data modifications and recalculations are easier to control. To use this method, you must define a currency partition (see "Main Database" on page 217).

Either of these two methods may require a currency conversion to be applied at report time. Report time conversion enables analysis of various exchange rate scenarios without actually storing data in the database. The currency conversion module enables performance of ad hoc conversions. You perform ad hoc conversions by using Spreadsheet Add-in, as discussed in the *Essbase Spreadsheet Add-in User's Guide*, or by using a report script, as discussed under "Converting Currencies in Report Scripts" on page 228.

Building Currency Conversion Applications and Performing Conversions

To build a currency conversion application and perform conversions, use the following process:

1. Create or open the main database outline. See "Creating Main Database Outlines" on page 223.
2. Prepare the main database outline for currency conversion. See "Preparing Main Database Outlines" on page 223.

3. Generate the currency database outline. See "Generating Currency Database Outlines" on page 224.
4. Link the main and currency databases. See "Linking Main and Currency Databases" on page 224.
5. Convert currency values. See "Converting Currency Values" on page 224.
6. Track currency conversions. See "Tracking Currency Conversions" on page 229.
7. If necessary, troubleshoot currency conversion. See "Troubleshooting Currency Conversion" on page 231.

Creating Main Database Outlines

To create a main database outline, you need to create or open an Analytic Services database outline, modify the outline as needed, and then save the outline for use in the currency conversion application.

➤ To create a new outline or open an existing outline, use any of the following methods:

Tool	Topic	Location
Administration Services	Opening and Editing Outlines	*Essbase Administration Services Online Help*
MaxL	**create database**	*Technical Reference*
ESSCMD	CREATEDB	*Technical Reference*

Preparing Main Database Outlines

After you create or open the main database outline, you need to modify dimensions and members to enable Analytic Services to generate the currency database outline automatically. For more information, see "Main Database" on page 217.

➤ To prepare a main database outline, see "Preparing the Main Database Outline for Currency Conversion" in *Essbase Administration Services Online Help*.

Generating Currency Database Outlines

After you verify and save the main database outline, you can generate the currency outline. The currency outline contains dimensions, members, currency names, and currency categories previously defined in the main database outline. The currency database outline is basically structured and ready to use after being generated but may require additions to make it complete.

➤ To generate a currency database outline, see "Generating a Currency Database Outline" in *Essbase Administration Services Online Help*.

Linking Main and Currency Databases

To perform a currency conversion calculation, Analytic Services must recognize a link between the main and currency databases. Generating a currency outline does not automatically link a main database with a currency database. When you link the databases, you specify the conversion calculation method and the default currency type member.

➤ To link main and currency databases, see "Linking a Database to a Currency Database" in *Essbase Administration Services Online Help*.

Converting Currency Values

After you create a currency conversion application, you convert data values from a local currency to a common, converted currency by using the CCONV command in calculation scripts. For example, you might convert data from a variety of currencies into USD (U.S. dollars). You can convert the data values back to the original, local currencies by using the CCONV TOLOCALRATE command.

You can convert all or part of the main database using the rates defined in the currency database. You can overwrite local values with converted values, or you can keep both local and converted values in the main database, depending on your tracking and reporting needs.

Note: When running a currency conversion, ensure that the data being converted is not simultaneously being updated by other user activities (for example, a calculation, data load, or currency conversion against the same currency partition). Concurrent activity on the data being converted may produce incorrect results. Analytic Services does not display a warning message in this situation.

Note: When you convert currencies using the CCONV command, the resulting data blocks are marked as dirty for the purposes of Intelligent Calculation. Thus, Analytic Services recalculates all converted blocks when you recalculate the database.

To see sample currency conversion calculation scripts, see the *Technical Reference*.

Overwriting Local Values with Converted Values

If you want to overwrite local values, you do *not* need to create a currency partition dimension in the main database. Use the CCONV command in a calculation script to convert all data in the database:

The following calculation script converts the values in the database to USD:

```
CCONV USD;
CALC ALL;
```

If required, you can specify a currency name that contains the required exchange rate. The following calculation script converts the values in the database to USD, using the exchange rate for Jan as defined in the currency database:

```
CCONV Jan->USD;
CALC ALL;
```

The CALC ALL command is required in the examples shown because the CCONV command only converts currencies. It does not consolidate or calculate members in the database.

The following calculation script uses the “Act xchg” rate to convert the converted values back to their original local currency values:

```
CCONV TOLOCALRATE "Act xchg";
CALC ALL;
```

Note: You cannot use the FIX command unless you are using a currency partition dimension and the CCTRACK setting is TRUE in the `essbase.cfg` file.

Keeping Local and Converted Values

You can keep both local and converted values in a database. In the main database you need to define the members that store the local and the converted values. You define the members by creating a currency partition dimension (see “Main Database” on page 217). The currency partition dimension has two partitions, one for local values and one for converted values.

➤ To create a calculation script that copies local data to a converted partition and calculates the data, use the following process:

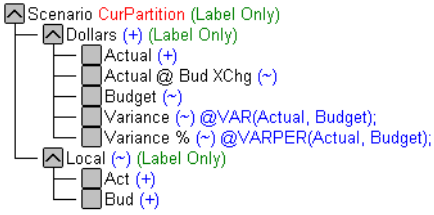
1. Use the DATACOPY command to copy data from the local to the converted partition.
2. Use the FIX command to calculate only the converted partition and use the CCONV command to convert the data.

   Note: When using a currency partition dimension, you must FIX on a member of the dimension to use the CCONV command.

3. Use the CALC command to recalculate the database.

The following example is based on the Sample Interntl database and the corresponding Sample Xchgrate currency database. Figure 67 shows the currency partition from the Sample Interntl database.

Figure 67: Calculating Local and Converted Currency Conversions



The following calculation script performs three currency conversions for Actual, Budget, and Actual @ Bud Xchg data values:

```
/* Copy data from the local partition to the master partition
(for converted values) */
DATACOPY Act TO Actual;
DATACOPY Bud TO Budget;

/* Convert the Actual data values using the "Act xchg" rate */

FIX(Actual)
     CCONV "Act xchg"->US$;
ENDFIX

/* Convert the Budget data values using the "Bud xchg" rate */
FIX(Budget)
     CCONV "Bud xchg"->US$;
ENDFIX
```

```
/* Convert the "Actual @ Bud XChg" data values using the
"Bud xchg" rate */
FIX("Actual @ Bud XChg")
     CCONV "Bud xchg"->US$;
ENDFIX

/* Recalculate the database */
CALC ALL;
CALC TWOPASS;
```

The following calculation script converts the Actual and Budget values back to their original local currency values:

```
FIX(Actual)
CCONV TOLOCALRATE "Act xchg";
ENDFIX
FIX(Budget)
CCONV TOLOCALRATE "Bud xchg";
ENDFIX
CALC ALL;
```

Note: When you convert currencies using the CCONV command, the resulting data blocks are marked as dirty for the purposes of Intelligent Calculation. Thus, Analytic Services recalculates all converted blocks when you recalculate the database.

Calculating Databases

If you execute a CALC ALL command to consolidate the database after running a conversion, meaningful total-level data is generated in the converted base rate partition, but the local rate partition contains a meaningless consolidation of local currency values. To prevent meaningless consolidation, use the calculation command SET UPTOLOCAL, which restricts consolidations to parents with the same defined currency. For example, all cities in the US use dollars as the unit of currency. Therefore, all children of US consolidate to US. Consolidation stops at the country level, however, because North America contains countries that use other currencies.

Converting Currencies in Report Scripts

You can convert currencies in report scripts, using the CURRENCY command to set the output currency and the currency type. For the syntax and definitions of Report Writer commands, see the *Technical Reference*.

Note: Analytic Services cannot perform “on the fly” currency conversions across transparent databases. If you have two transparent partition databases that are calculated using different conversions, you cannot perform currency conversions in reports.

The following Sample report contains first quarter Budget Sales for colas, using the January exchange rate for the Peseta currency.

```
                        Illinois Sales Budget

                      Jan      Feb      Mar
                      ======== ======== ========
100-10                3           3        3
100-20                2           2        2
100-30                #Missing #Missing #Missing
100                   5           5        5
            Currency: Jan->Peseta->Act xchg

            Currency: Jan->Peseta->Act xchg
                      Illinois Sales Budget
                      Jan      Feb      Mar
                      ======== ======== ========
100-10                3           3        3
100-20                2           2        2
100-30                #Missing #Missing #Missing
100                   5           5        5
```

Use the following script to create the Sample currency conversion report:

```
<Page (Market, Measures, Scenario)
{SupCurHeading}
Illinois Sales Budget
       <Column (Year)
       <children Qtr1
<Currency "Jan->Peseta->Act xchg"
<Ichildren Colas
   !
```

```
{CurHeading}
Illinois Sales Budget
       <Column (Year)
       <children Qtr1
   !
```

Tracking Currency Conversions

You can use the CCTRACK setting in the `essbase.cfg` file to control whether Analytic Services tracks the currency partitions that have been converted and the exchange rates that have been used for the conversions. Tracking currency conversions has the following advantages:

- Enables conversion to occur at report time through Spreadsheet Add-in or Report Writer
- Enables conversion of a converted currency back to its original, local rate through use of the CCONV TOLOCALRATE command
- Prevents data inaccuracies due to accidental reconversion of data during a currency calculation.

By default CCTRACK is turned on. Analytic Services tracks which currency partitions have been converted and which have not. The tracking is done at the currency partition level: a database with two partitions has two flags, each of which can be either “converted” or “unconverted.” Analytic Services does not store a flag for member combinations within a partition. When CCTRACK is turned on, the following restrictions apply:

- If you are using currency partitions, you can use a FIX statement with the CCONV command only on currency partition members. For example, in the Sample Basic database, the following example is valid:

  ```
  FIX(Actual)
  CCONV "Act xchg"->US$;
  ENDFIX]
  ```

- If you are not using currency partitions, you must use CCONV with a FIX statement.

Reasons to Turn Off CCTRACK

For increased efficiency when converting currency data between currency partitions, you may want to turn off CCTRACK. For example, you load data for the current month into the local partition, use the DATACOPY command to copy the entire currency partition that contains the updated data, and then run the conversion on the currency partition.

Note: Always do a partial data load to the local partition and use the DATACOPY command to copy the *entire* currency partition to the converted partition before running the currency conversion. Updating data directly into the converted partition causes incorrect results.

Methods for Turning Off CCTRACK

You can turn off CCTRACK in three ways:

- Use the SET CCTRACKCALC ON|OFF command in a calculation script to turn off CCTRACK temporarily. You can use this command at calculation time to provide increased flexibility and efficiency during currency conversion.
- Use the CLEARCCTRACK calculation command to clear the internal exchange rate tables created by CCTRACK. You can use the command inside a FIX statement to clear the exchange rates for a currency partition. Use the command after a data load to reset the exchange rate tables so they are ready for future currency conversion calculations.
- Set CCTRACK to FALSE in the `essbase.cfg` file. Setting CCTRACK to False turns off the tracking system and has the following results:
  - The CCONV command assumes that data is unconverted (is in local currency). If you accidentally run the CCONV command multiple times on the same data, the resulting data is inaccurate.
  - Similarly, the currency report options assume that the data is unconverted (is in local currency). If the data has already been converted in the database, it is reconverted at report time, resulting in inaccurate data.
  - The restrictions on using the FIX and DATACOPY commands in currency conversions do not apply.

Note: When running a currency conversion, ensure that the data being converted is not simultaneously being updated by other user activities (for example, a calculation, data load, or currency conversion against the same currency partition). Concurrent activity on the data being converted may produce incorrect results. Analytic Services does not display a warning message in this situation.

Troubleshooting Currency Conversion

For information about how to troubleshoot currency conversions, see "Troubleshooting Currency Conversion" in *Essbase Administration Services Online Help*.

Chapter 13

Designing Partitioned Applications

An Analytic Services partitioned application can span multiple servers, processors, or computers. A partition is the piece of a database that is shared with another database. Partitioning applications can provide the following benefits:

- Improved scalability, reliability, availability, and performance of databases
- Reduced database sizes
- More efficient use of resources

Note: The information in this chapter is designed for block storage databases. Some of the information is not relevant to aggregate storage databases. For detailed information on the differences between aggregate and block storage, see Chapter 57, "Comparison of Aggregate and Block Storage." For information on creating aggregate storage applications, see Chapter 58, "Aggregate Storage Applications, Databases, and Outlines."

This chapter contains the following sections:

CAUTION: Design partitions carefully. Hyperion strongly recommends that you read this chapter before creating partitions.

Process for Designing a Partitioned Database

Here is the suggested process for designing a partitioned database.

1. Learn about partitions. See "Understanding Analytic Services Partitioning" on page 234.
2. Determine whether the database can benefit from partitioning. See "Deciding Whether to Partition a Database" on page 239.
3. Identify the data to partition. See "Determining Which Data to Partition" on page 241.
4. Decide on the type of partition. See "Deciding Which Type of Partition to Use" on page 242.
5. Understand the security issues related to partitions. See "Planning for Security for Partitioned Databases" on page 261.

Understanding Analytic Services Partitioning

Analytic Services Partitioning is a collection of features that makes it easy to design and administer databases that span Analytic Services applications or servers. Partitioning is licensed separately from Analytic Services. The Partitioning option must be licensed for every server that contains a database partition.

Partitioning can provide the following benefits:

- Synchronize the data in multiple partitioned databases. Analytic Services tracks changes made to data values in a partition and provides tools for updating the data values in related partitions.
- Synchronize the outlines of multiple partitioned databases. Analytic Services tracks changes made to the outlines of partitioned databases and provides tools for updating related outlines.
- Allow users to navigate between databases with differing dimensionality. When users drill across to the new database, they can drill down to more detailed data.

Based on user requirements, select one of the following partitioning strategies:

- Partition applications from the top down. Use *top-down partitioning* to split a database onto multiple processors, servers, or computers. Top-down partitioning can improve the scalability, reliability, and performance of databases. To achieve the best results with top-down partitioning, create a separate application for each partitioned database.
- Partition applications from the bottom up. Use *bottom-up partitioning* to manage the flow of data between multiple related databases. Bottom-up partitioning can improve the quality and accessibility of the data in databases.
- Partition databases according to attribute values associated with base dimensions (a base dimension is a standard dimension associated with one or more attribute dimensions). Partitioning a base dimension according to its attributes enables the user to extract data based on the characteristics of a dimension such as flavor or size.

This section contains the following sections:

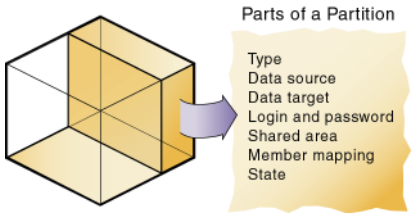
What Is a Partition?

A *partition* is a piece of a database that is shared with another database. Partitions contain the following parts, as illustrated in Figure 68.

- Type of partition. A flag indicating whether the partition is replicated, transparent, or linked.
- Data source information. The server, application, and database name of the data source.
- Data target information. The server, application, and database name of the data target.
- Login and password. The login and password information for the data source and the data target. This information is used for internal requests between the two databases to execute administrative and user operations.

- Shared areas. A definition of one or more areas, or subcubes, shared between the data source and the data target. To share more than one non-contiguous portion of a database, define multiple areas in a single partition. This information determines which parts of the data source and data target are shared so that Analytic Services can put the proper data into the data target and keep the outlines for the shared areas synchronized.
- Member mapping information. A description of how the members in the data source map to members in the data target. Analytic Services uses this information to determine how to put data into the data target if the data target and the data source use different names for some members and dimensions.
- State of the partition. Information about whether the partition is up-to-date and when the partition was last updated.

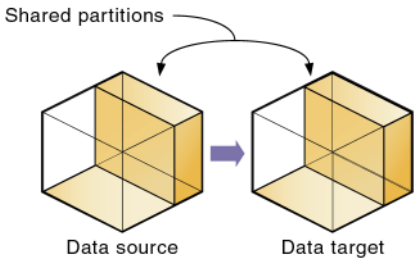
Figure 68: Parts of a Partition



Data Sources and Data Targets

Partitioned databases contain at least one *data source*, the primary site of the data, and at least one *data target*, the secondary site of the data. A single database can serve as both the data source for one partition and the data target for another. When you define a partition, you map cells in the data source to their counterparts in the data target:

Figure 69: Data Source and Data Target



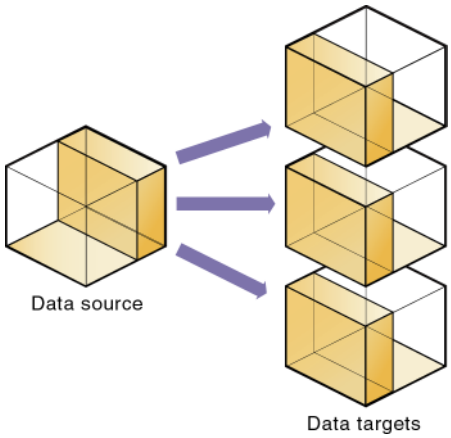
An Analytic Services database can contain many different partitions as well as data that is not shared with any other Analytic Services database. You can define partitions between the following databases:

- Different databases in different applications, as long as each database uses the same language (for example, German).
- Different databases in different applications on different processors or computers, as long as each database uses the same language (for example, German).
- Different databases in one application. This practice is not recommended, because you cannot reap the full benefits of partitioning databases unless each database is in a separate application.

You can only define one partition of each type between the same two databases. For example, you can only create one replicated partition between the Sampeast East database and the Samppart Company database. The East or Company databases can, however, contain many replicated partitions that connect to other databases.

A single database can serve as the data source or data target for multiple partitions. To share data among many databases, create multiple partitions, each with the same data source and a different data target:

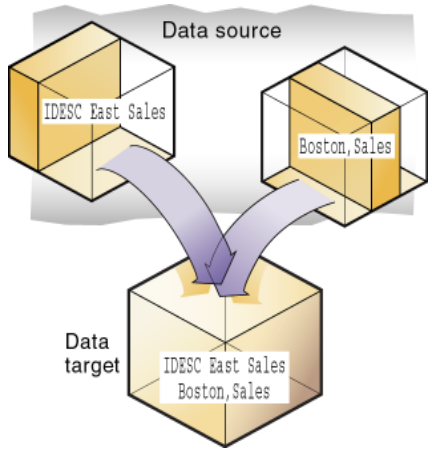
Figure 70: Data Shared at Multiple Targets



Overlapping Partitions

An overlapping partition occurs when similar data from two or more databases serve as the data source for a single data target in a partition. For example, `IDESC East, Sales` from database 1 and `Boston, Sales` from database 2 are mapped to `IDESC East, Sales` and `Boston, Sales` in database 3. Because `Boston` is a member of the dimension `East`, the data for `Boston` mapped to database 3 from database 1 and database 2, overlap. This data overlap results in an overlapping partition:

Figure 71: Overlapping Partitions



An overlapping partition is allowed in linked partitions, but is invalid in replicated and transparent partitions and generates an error message during validation.

Attributes in Partitions

You can use attribute functions for partitioning on attribute values. But you cannot partition an attribute dimension. Use attribute values to partition a database when you want to access members of a dimension according to their characteristics.

For example, in the Sample Basic database, you cannot partition the Pkg Type attribute dimension. But you can create a partition that contains all the members of the Product dimension that are associated with either or both members (Bottle and Can) of the Pkg Type dimension. If you create a partition that contains members associated with Can, you can access data only on Product members that are packaged in cans; namely, 100-10, 100-20, and 300-30.

You can use the @ATTRIBUTE command and the @WITHATTR command to define partitions.

For example, to extract data on all members of the Product dimension that are associated with the Caffeinated attribute dimension, you can create a partition such as @ATTRIBUTE (Caffeinated). But you cannot partition the Caffeinated attribute dimension.

Based on the previous example, this partition is correct:

Figure 72: Correct Partitioning

```
Source                    Target
@ATTRIBUTE(Caffeinated)   @ATTRIBUTE(Caffeinated)
```

Based on the previous example, this partition is incorrect:

Figure 73: Incorrect Partitioning

```
Source                    Target
Caffeinated               Caffeinated
```

For more information about these commands, refer to the section on calculation commands in the *Technical Reference*.

For more information on attribute dimensions, see Chapter 10, “Working with Attributes.”

Deciding Whether to Partition a Database

Partitioning a database is not always the correct option. The following sections provide questions you can use to determine if partitioning the database is the best solution for you.

When to Partition a Database

Review the following list of questions. If you answer yes to many of them, or answer yes to some that are very important to you, you may wish to partition databases.

- Should the data be closer to the people who are using it? Is the network being stressed because users are accessing data that is far away?
- Would a single failure be catastrophic? If everyone is using a single database for mission-critical purposes, what happens if the database goes down?
- Does it take too long to perform calculations after new data is loaded? Can you improve performance by spreading the calculations across multiple processors or computers?
- Do users want to see the data in different application contexts? Would you like to control how they navigate between databases?
- Do you have separate, disconnected databases storing related information? Does the related information come from different sources? Are you having trouble synchronizing it?
- Will you add many new organizational units? Would they benefit from having their own databases? Partitioned databases help you grow incrementally.
- Are users having to wait as other users access the database?
- Do you want to save disk space by giving users access to data stored in a remote location?
- Should you reduce network traffic by replicating data in several locations?
- Do you need to control database outlines from a central location?

When Not to Partition a Database

Sometimes, it does not make sense to partition a centralized database. Partitioning a database can require additional disk space, network bandwidth, and administrative overhead. Review the following list of questions. If you answer yes to many of them, or answer yes to some that are very important to you, you may not want to partition a database.

- Do you have resource concerns? For example, are you unable to purchase more disk space or allow more network traffic?

- Do you perform complex allocations where unit level values are derived from total values?
- Are you required to keep all databases online at all times? Keeping databases online can be a problem if you have databases in several time zones, because peak user load may differ between time zones. Using linked and transparent partitions exacerbate this problem, but using replicated partitions might help.
- Are the databases in different languages? Analytic Services can only partition databases if both databases use the same language, such as German.

Determining Which Data to Partition

When designing a partitioned database, find out the following information about the data in the database:

- Which database should be the data source and which the data target? The database that "owns" the data should be the data source. Owning the data means that this is the database where the data is updated and where most of the detail data is stored.
- Are some parts of the database accessed more frequently than others?
- What data can you share among multiple sites?
- How granular does the data need to be at each location?
- How frequently is the data accessed, updated, or calculated?
- What are the available resources? How much disk space is available? CPUs? Network resources?
- How much data needs to be transferred over the network? How long does that take?
- Where is the data stored? Is it in one location or in more than one location?
- Where is the data accessed? Is it in one location or in more than one location?
- Is there information in separate databases that should be accessed from a central location? How closely are groups of data related?

The answers to these questions determine which data to include in each partition. For examples, see "Case Studies for Designing Partitioned Databases" on page 262.

Note: You cannot partition attribute dimensions. See "Attributes in Partitions" on page 238.

Deciding Which Type of Partition to Use

Analytic Services supports the following types of partitions:

- A *replicated partition* is a copy of a portion of the data source that is stored in the data target.
- A *transparent partition* allow 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 Analytic Services database, or on another Analytic Server.
- A *linked partition* sends users from a cell in one database to a cell in another database. Linked partitions give users a different perspective on the data.

Replicated Partitions

A replicated partition is a copy of a portion of the data source that is stored in the data target. Some users can then access the data in the data source while others access it in the data target.

In the Samppart and Sampeast applications shipped with Analytic Services, for example, the database administrator at The Beverage Company (TBC) created a replicated partition between the East database and the Company database containing Actual, Budget, Variance, and Variance%. Users in the eastern region now store their budget data locally. Because they do not have to retrieve this data live from the corporate headquarters, their response times are faster and they have more control over the down times and administration of the local data. For a more complete description of the sample partitioned databases provided with Analytic Services, see "Case Study 1: Partitioning an Existing Database" on page 262.

Changes to the data in a replicated partition flow from the data source to the data target. Changes made to replicated data in the data target do not flow back to the data source. If users change the data at the data target, Analytic Services overwrites their changes when the database administrator updates the replicated partition.

The database administrator can prevent the data in the replicated portion of the data target from being updated. This setting takes precedence over access provided by security filters and is also honored by batch operations such as dataload and calculation. By default, replicated partitions are not updateable. For directions on how to set a partition as updateable, see the *Essbase Administration Services Online Help*.

Use a replicated partition when you want to achieve any of the following goals:

- Decrease network activity.
- Decrease query response times.
- Decrease calculation times.
- Make it easier to recover from system failures.

These sections help you assess the value of replicated partitions:

- "Rules for Replicated Partitions" on page 243
- "Advantages and Disadvantages of Replicated Partitions" on page 245
- "Performance Considerations for Replicated Partitions" on page 246
- "Replicated Partitions and Port Usage" on page 247

Rules for Replicated Partitions

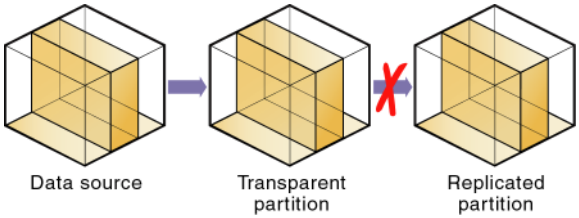
Replicated partitions must follow these rules:

- You must be able to map the shared replicated areas of the data source and data target outlines even though the shared areas do not have to be identical. You must tell Analytic Services how each dimension and member in the data source maps to each dimension and member in the data target.

  The data source and data target outlines for the non-shared areas do not have to be mappable.

- You cannot create a replicated partition on top of a transparent partition. In other words, none of the areas that you use as a replicated partition target can come from a transparent partition source:

  Figure 74: Invalid Replicated Partition

- The cells in the data target of a replicated partition cannot come from two different data sources; the cells in one partition must come from just one database. If you want to replicate cells from more than one database, create a different partition for each data source.

  The cells in a data target can be the data source for a different replicated partition. For example, if the Samppart Company database contains a replicated partition from the Sampeast East database, you can replicate the cells in the Sampeast East database into a third database, such as the Sampwest West database.

- You cannot use attribute members to define a replicated partition. For example, associated with the Market dimension, the Market Type attribute dimension members are Urban, Suburban, and Rural. You cannot define a partition on Urban, Suburban, or Rural because a replicated partition contains dynamic data, not stored data. Hence, an attempt to map attributes in replicated partitions results in an error message. However, you can use the WITHATTR command to replicate attribute data.

For a discussion of using Dynamic Time Series members in replicated partitions, see “Using Dynamic Time Series Members in Partitions” on page 578.

Advantages and Disadvantages of Replicated Partitions

Replicated partitions can solve many database problems, but replicated partitions are not always the ideal partition type. This section describes the advantages and disadvantages of using a replicated partition.

Advantages of Replicated Partitions

Following are the advantages of using a replicated partition.

- Replicated partitions can decrease network activity, because the data is now stored closer to the end users, in the data target. Decreased network activity results in improved retrieval times for the users.
- The data is more easily accessible to all users. Some users access the data at the data source, others at the data target.
- Failures are not as catastrophic. Because the data is in more than one place, if a single database fails, only the users connected to that database are unable to access the information. It is still available at and can be retrieved from the other sites.
- Local database administrators can control the down time of their local databases. For example, because users in the eastern region are accessing their own replicated data instead of the Company database, the database administrator can bring down the Company database without affecting the users in the eastern region.
- Because only the relevant data is kept at each site, databases can be smaller. For example, users in the eastern region can replicate just the eastern budget information, instead of accessing a larger company database containing budget information for all regions.

Disadvantages of Replicated Partitions

Following are the disadvantages of using a replicated partition.

- You need more disk space, because you are storing the data in two or more locations.
- The data must be refreshed regularly by the database administrator, so it is not up-to-the-minute.

Performance Considerations for Replicated Partitions

To improve the performance of replicated partitions, consider the following when replicating data.

- Do not replicate members that are dynamically calculated in the data source to greatly reduce replication time, because Analytic Services must probe the outline to find dynamically calculated members and their children to determine how to perform the calculation.
- Do not replicate derived data from the data source to greatly reduce replication time. Instead, try to replicate the lowest practical level of each dimension and perform the calculations on the data target after you complete the replication.

  For example, to replicate the database along the Market dimension:

  – Define the shared area as the lowest level members of the Market dimension that you care about, for example, East, West, South, and Central and the level 0 members of the other dimensions.

  – After you complete the replication, calculate the values for Market and the upper level values in the other dimensions at the data target.

  Sometimes you cannot calculate derived data at the data target. In that case, you must replicate it from the data source. For example, you cannot calculate derived data at the data source if the data meets any of the following criteria:

  – Requires data outside the replicated area to be calculated.

  – Requires calculation scripts from which you cannot extract just the portion to be calculated at the data target.

  – Is being replicated onto a computer with little processing power, such as a laptop.
- Partitioning along a dense dimension takes more time than partitioning along a sparse dimension. When Analytic Services replicates data partitioned along a dense dimension, it must access every block in the data source and then create each block in the data target during the replication operation. For example, if the Market dimension were dense and you replicated the data in the East member, Analytic Services must access every block in the database and then create each block at the data target during the replication operation.

- You cannot replicate data into a member that is dynamically calculated at the data target. Dynamic Calc and Dynamic Calc and Store members do not contain any data until a user requests the data at run time. Analytic Services does not load or replicate into Dynamic Calc and Dynamic Calc and Store members. Analytic Services avoids sending replicated data for both dynamic dense and dynamic sparse members on the replication target, since this data is not stored on the data target.
- See "Populating or Updating Replicated Partitions" on page 289 to replicate only the data values that have changed instead of the entire partition.

Replicated Partitions and Port Usage

One port is used for every unique user and computer combination. If a user defines several replicated partitions on one server using the same user name, then only one port is occupied.

In a replicated partition, when a user (user1) drills into an area in the target that accesses source data, user1 is using the user name declared in the partition definition (partition user) to access the data from the source database. This access causes the use of an additional port because different users (user1 and partition user) are connecting to the application.

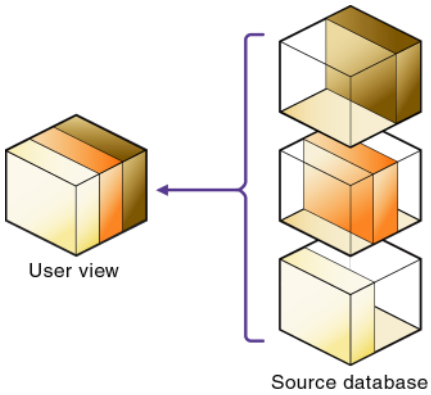
If a second user (user2) connects to the target database and drills down to access source data, user2 also uses the user name declared in the partition definition (partition user) to access the source database. Because the partition user is already connected to the source database, an additional port is not needed for the partition user, as long as user2 is accessing the same source database.

Note: Because of the short-term nature of replication, replicated partitions and ports are rarely a problem.

Transparent Partitions

A transparent partition allows users to manipulate data that is stored remotely as if it were part of the local database. The remote data is retrieved from the data source each time that users at the data target request it. Users do not need to know where the data is stored, because they see it as part of their local database.

Figure 75: Transparent Partitions



Because the data is retrieved directly from the data source, users see the latest version of the data. When they update the data, their updates are written back to the data source. This process means that other users at both the data source and the data target have immediate access to those updates.

With a transparent partition, users at the data source may notice slower performance as more users access the source data and users at the data target may notice slower performance as more users access the source data.

For example, the database administrator at TBC can use a transparent partition to calculate each member of the Scenario dimension on a separate CPU. This process reduces the elapsed time for the calculation, while still providing users with the same view of the data. For a more complete description of a transparent partition based on the Sample Basic database, see "Case Study 1: Partitioning an Existing Database" on page 262.

Use a transparent partition when you want to achieve the following goals:

- Show users the latest version of the data.
- Allow users at the data target to update data.
- Decrease disk space.

These sections help you assess the value of transparent partitions:

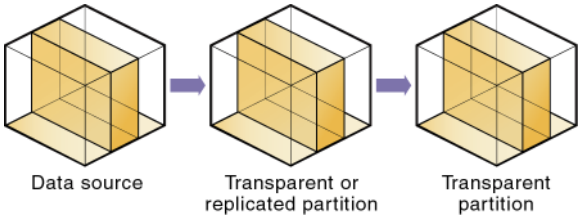
Rules for Transparent Partitions

Transparent partitions must follow these rules:

- The shared transparent areas of the data source and data target outlines do not have to be identical, but you must be able to map the dimensions in them. You must tell Analytic Services how each dimension and member in the data source maps to each dimension and member in the data target.
- The data source and data target outlines for the non-shared areas do not have to be mappable, but attribute associations should be identical. Otherwise, users can get incorrect results for some retrievals. For example, if product 100-10-1010 is associated with the Grape Flavor attribute on the source, but product 100-10-1010 is not associated with Grape on the target, the total of sales for all Grape flavors in New York is incorrect.
- You cannot use attribute dimensions or members to define a transparent partition. For example, associated with the Market dimension, the Market Type attribute dimension has members Urban, Suburban, and Rural. You cannot define a partition on Urban, Suburban, or Rural.

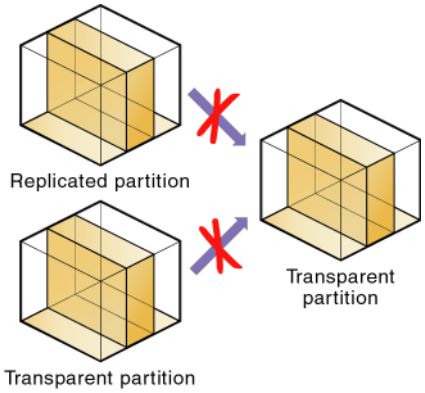
- You can create a transparent partition on top of a replicated partition. In other words, you can create a transparent partition target using a replicated partition source:

  Figure 76: Valid Transparent Partition

- As illustrated in Figure 77, you cannot create a transparent partition on top of more than one other partition. In other words, you cannot create a transparent partition target from multiple sources because each cell in a database must be retrieved from only one location—either the local disk or a remote disk.

  Figure 77: Invalid Transparent Partition

- Carefully consider any formulas you assign to members in the data source and data target.

For a discussion on using Dynamic Time Series members in transparent partitions, see "Using Dynamic Time Series Members in Partitions" on page 578.

Advantages and Disadvantages of Transparent Partitions

Transparent partitions can solve many database problems, but transparent partitions are not always the ideal partition type. This section describes the advantages and disadvantages of using a transparent partition.

Advantages of Transparent Partitions

Following are the advantages of using a transparent partition:

- You need less disk space, because you are storing the data in one database.
- The data accessed from the data target is always the latest version.
- When the user updates the data at the data source, Analytic Services makes those changes at the data target.
- Individual databases are smaller, so they can be calculated more quickly.
- The distribution of the data is invisible to the end user and the end user's tools.
- You can load the data from either the data source or the data target.

Disadvantages of Transparent Partitions

Following are the disadvantages of using a transparent partition:

- Transparent partitions increase network activity, because Analytic Services transfers the data at the data source across the network to the data target. Increased network activity results in slower retrieval times for users.
- Because more users are accessing the data source, retrieval time may be slower.
- If the data source fails, users at both the data source and the data target are affected. Therefore, the network and data source must be available whenever users at the data source or the data target need them.
- You can perform some administrative operations only on local data. For example, if you archive the data target, Analytic Services archives just the data target and does *not* archive the data source. The following administrative operations work only on local data:
  - CLEARDATA calculation command
  - DATACOPY calculation command

  - EXPORT command
  - VALIDATE command
  - BEGINARCHIVE and ENDARCHIVE commands
  - Restructure operations in Administration Services

- When you perform a calculation on a transparent partition, Analytic Services performs the calculation using the current values of the local data and transparent dependents. Analytic Services does not recalculate the values of transparent dependents. To calculate all partitions, issue a CALC ALL command for each individual partition, and then perform a CALC ALL command at the top level using the new values for each partition.

  Analytic Services does not recalculate the values of transparent dependents because the outlines for the data source and the data target may be so different that such a calculation is accurate.

  For example, suppose that the data target outline contained a Market dimension with East, West, South, and Central members and the data source outline contained an East dimension with New York and New Jersey members. If you tried to calculate the data target outline, you would assume that East was a level 0 member. In the data source, however, East is derived by adding New York and New Jersey. Any calculations at the data target, however, would not know this information and could not reflect any changes made to New York and New Jersey in the data source. To perform an accurate calculation, therefore, you must first calculate East in the data source and then calculate the data target.

  For a tip on improving performance of transparent calculations, see "Calculating Transparent Partitions" on page 253.

- Formulas assigned to members in the data source may produce calculated results that are inconsistent with formulas or consolidations defined in the data target, and vice versa.

If these disadvantages are too serious, consider using replicated or linked partitions instead.

Performance Considerations for Transparent Partitions

To improve the performance of transparent partitions, consider the following facts when creating the partition:

- Partitioning along dense dimensions in a transparent partition can greatly slow performance. The slow performance results because dense dimensions are used to determine the structure and contents of data blocks. If a database is partitioned only along a dense dimension at the target, Analytic Services must compose data blocks by performing network calls for the remote data in the transparent partition in addition to the disk I/O for the local portion of the block. To improve performance, consider including one or more sparse dimensions in the area definition so that the number of blocks required is limited to combinations with the sparse members.
- Basing transparent partitions on the attribute values of a dimension can increase retrieval time, because attributes are associated with sparse dimensions. In such cases, partitioning at a level higher than the level that is associated with attributes improves retrieval time. For example, in the Product dimension of the Sample Basic database, if children 100-10, 200-10, and 300-10 (level 0) are associated with attributes, then partition their parents 100, 200, and 300 (level 1) for better retrieval performance.
- Loading data into the data source from the data target can greatly slow performance. If possible, load data into the data source locally.
- Retrieval time is slower because users access the data over the network.
- Partitioning base dimensions can greatly slow performance.
- For calculation-related performance considerations, see “Performance Considerations for Transparent Partition Calculations” on page 254.

Calculating Transparent Partitions

When you perform a calculation on a transparent partition, Analytic Services performs the calculation using the current values of the local data and transparent dependents. When calculating local data that depends on remote data, Analytic Services performs a bottom-up calculation. The bottom-up calculation can be done only if the calculator cache on the target database is used properly. For complete information on bottom-up calculations, see “Using Bottom-Up Calculation” on page 1200. For information on the calculator cache, see “Sizing the Calculator Cache” on page 1133.

Increasing the amount of memory assigned to the calculator cache greatly improves calculation performance with transparent partitions. When a calculation is started, a message in the application log indicates whether or not the calculator cache is enabled or disabled on the target database. Using the calculator cache on the target database reduces the number of blocks that are requested from the data source during calculation. Reducing the blocks requested, in turn, reduces the amount of network traffic that is generated by transferring blocks across the network. For information on estimating the size of the calculator cache, see "Sizing the Calculator Cache" on page 1133.

Performance Considerations for Transparent Partition Calculations

Calculating data on the data target can greatly slow performance when the data target must retrieve each dependent data block across the network, and then perform the calculation.

Performance with transparent calculations may also slow if Analytic Services must perform a top-down calculation on any portion of the data target that contains top-down member formulas. When the data target contains no top-down member formulas, Analytic Services can perform a bottom-up calculation on the data target, which is much faster.

When Analytic Services performs the calculation on the data source, it can always perform a bottom-up calculation. For a comparison of top-down and bottom-up calculations, see "Using Bottom-Up Calculation" on page 1200.

Consider using these calculation alternatives:

- Dynamic Calc or Dynamic Calc and Store members as parents of the transparent data so that the data is calculated on the fly when it is retrieved. This process reduces the batch processing time for batch calculation. Analytic Services performs the calculation only when users request it.
- A replicated layer between the low-level transparent data and high-level local data.

Other performance strategies include the following:

- Keep the partition fully within the calculator cache area (see "Sizing the Calculator Cache" on page 1133). Keeping a partition fully within the calculator cache means that any sparse members in the partition definition

must be contained within the calculator cache. For example, in the Sample Basic database, if a partition definition includes @IDESC(East), all descendants of East must be within the calculator cache.

- Enable the calculator cache, and assign a sufficient amount of memory to it.
- Do not use complex formulas on any members that define the partition. For example, in Sample Basic, assigning a complex formula to New York or New Jersey (both children of East) forces Analytic Services to use the top-down calculation method. For more information, see "Bottom-Up and Top-Down Calculation" on page 1200.

Transparent Partitions and Member Formulas

If the data target and data source outlines are identical except for different member formulas, make sure that the partition definition produces the desired calculation results.

For example, suppose that the data source and data target outlines both contain a Market dimension with North and South members, and children of North and South. On the data target, Market is calculated from the data for the North and South members (and their children) on the data source. If any of these members on the data source contain member formulas, these formulas are calculated, thus affecting the calculated value of Market on the data target. These results may be different from how the Market member are calculated from the North and South members on the data target, where these formulas may not exist.

Make sure that any formulas you assign to members in the data source and data target produce the desired results.

Transparent Partitions and Port Usage

One port is used for every unique user and machine combination. If a user defines several transparent partitions on one server, using the same user name, then only one port is occupied.

In a transparent partition, when a user (user1) drills into an area in the target that accesses source data, user1 is using the user name declared in the partition definition (partition user) to access the data from the source database. This process causes the use of an additional port because different users (user1 and partition user) are connecting to the application.

If a second user (user2) connects to the target database and drills down to access source data, user2 also uses the user name declared in the partition definition (partition user) to access the source database. Because the partition user is already connected to the source database, an additional port is not needed for the partition user, as long as user2 is accessing the same source database.

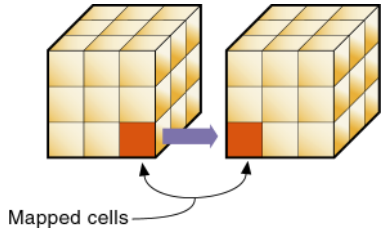
Linked Partitions

A linked partition connects two different databases with a data cell. When the end user clicks the linked cell in the data target, you drill across to a second database, the data source, and view the data there. If you are using Spreadsheet Add-in, for example, a new sheet opens displaying the dimensions in the second database. You can then drill down into these dimensions.

Unlike replicated or transparent partitions, linked partitions do not restrict you to viewing data in the same dimensionality as the target database. The database that you link to can contain very different dimensions than the database from which you connected. With linked partitions, data is not physically transferred from the source to the target. Instead, a data cell or range of cells on the target provides a link point to a cell or range of cells on the source.

To prevent users from seeing privileged data, establish security filters on both the data source and the data target. For directions on establishing security filters, see "Planning for Security for Partitioned Databases" on page 261.

Figure 78: Linked Partition



There are no performance considerations for linked partitions, beyond optimizing the performance of each linked database.

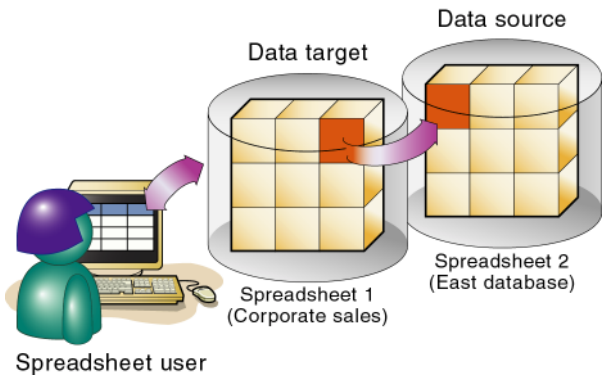
For example, if TBC grew into a large company, they might have several business units. Some data, such as profit and sales, exists in each business unit. TBC can store profit and sales in a centralized database so that the profit and sales for the entire company are available at a glance. The database administrator can link

individual business unit databases to the corporate database. For an example of creating a linked partition, see "Case Study 3: Linking Two Databases" on page 266.

A user in such a scenario can perform these tasks:

- View the general profit and sales at the corporate level in a spreadsheet at the data target.
- Drill across to individual business units, such as east. This action opens a new spreadsheet.
- Drill down in the new spreadsheet to more detailed data.

Figure 79: Source and Target for Linked Partition



For linked partitions, the spreadsheet that the user first views is connected to the data target, and the spreadsheet that opens when the user drills across is connected to the data source. This setup is the opposite of replicated and transparent databases, where users move from the data target to the data source.

Use a linked partition when you want to connect databases with different dimensionality.

These sections help you assess the value of linked partitions:

- "Advantages and Disadvantages of Linked Partitions" on page 258
- "Drill Across and Linked Partitions" on page 258
- "Linked Partitions and Port Usage" on page 259

Advantages and Disadvantages of Linked Partitions

Linked partitions allow users to navigate to databases that contain different dimensions, but linked partitions are not always the ideal partition type. This section describes the advantages and disadvantages of using a linked partition.

Advantages of Linked Partitions

Following are the advantages of linked partitions:

- You can view data in a different context; that is, you can navigate between databases containing many different dimensions.
- You do not have to keep the data source and data target outlines closely synchronized, because less of the outline is shared.
- A single data cell can allow the user to navigate to more than one database. For example, the Total Profit cell in the Accounting database can link to the Profit cells in the databases of each business unit.
- Performance may improve, because Analytic Services is accessing the database directly and not through a data target.

Disadvantages of Linked Partitions

Following are the disadvantages of linked partitions:

- You must create an account for users on each database or default access to the destination database (such as through a guest account). For information on creating accounts, see "Drill Across and Linked Partitions" on page 258.
- Users must access the linked database using Analytic Services Release 5-aware tools. If you have custom built tools, you must extend them using the Analytic Services Release 5 Grid API.

Drill Across and Linked Partitions

When a user clicks on a linked cell in a linked partition, a spreadsheet opens and displays the linked database. This process is called *drill across*. To facilitate drill across access you can use the following strategies:

- Create accounts for each user on each database. For example, if Mary accesses data in a Company database and an East database, create an account with the same login and password for Mary on both the Company and East databases. See "Managing Users and Groups" on page 845.

- Create a default account that users can use when accessing target databases. For example, if users access data through a data source named Company and a data target named East, create a guest account for the East database with the appropriate permissions. Once you have created the account, use the guest account login and password as the default login when creating the linked partition.

When a user drills across on data to a data target, Analytic Services logs the user into the data target using the following steps:

1. Checks to see if the user has an account on the data target with the same name and password. If so, Analytic Services logs the user in using that account.
2. Checks to see if you have specified a default account on the data target when you created the partition. If you did, Analytic Services logs the user in using that account.
3. Opens a login window prompting the user to enter a new login and password. Once the user enters a valid login and password, Analytic Services logs the user in using that account.

Linked Partitions and Port Usage

When accessing a linked partition, Analytic Services tries to use the end user's (user1) login information to connect to the source database. If user1 does not have access to the source database, Analytic Services looks for the linked partition default user name and password. If these defaults are not specified, user1 is requested to enter login information to access the source database. Port usage varies depending on the number of different user names being used to access the various source and target databases (and whether those databases are contained within the same or different servers).

Choosing a Partition Type

The following table should help you choose which type of partition to use.

Feature	Replicated	Transparent	Linked
Up-to-the-minute data		x	x
Reduced network traffic	x		x
Reduced disk space		x	x
Increased calculation speed	x		
Smaller databases		x	x
Improved query speed	x		x
Invisible to end users	x	x	
Access to databases with different dimensionality			x
Easier to recover	x		
Less synchronization required			x
Ability to query data based on its attributes		x	x
Ability to use front-end tools that are not Distributed OLAP-aware	x	x	
Easy to perform frequent updates and calculations		x	
Ability to update data at the data target		x	x
View data in a different context			x
Perform batch updates and simple aggregations	x		

Planning for Security for Partitioned Databases

Users accessing replicated, transparent, or linked partitions may need to view data stored in two or more databases. The following sections describe how to set up security so that users do not view or change inappropriate data.

Process for Setting up End User Security

Create the required end users with the correct filters.

1. Create accounts for users at the data target.

   See "Managing Users and Groups" on page 845.

2. Create read and write filters at the data target to determine what end users can view and update.

   See Chapter 36, "Managing Security for Users and Applications."

3. If you are creating a replicated partition, determine whether users can make changes to a replicated partition at the data target. This setting overrides user filters that allow users to update data.

   See the *Essbase Administration Services Online Help*.

4. If you are creating a linked partition, create accounts for users at the data source. Users accessing linked databases may need to connect to two or more databases.

   See "Drill Across and Linked Partitions" on page 258.

Process for Setting up Administrator Security

The administrative account performs all read and write operations requested by the data target for the data source. For example, when end users request data at the data target, the administrative account retrieves the data. When end users update data at the data target, the administrative account logs into the data source and updates the data there.

You can create filters on the administrative account in addition to filters on the end users. Filters on the administrative account can ensure that no one at the data target can view or update inappropriate data. For example, the administrator at the corporate database can restrict write access on certain cells to avoid relying on administrators in the various regions to set up security correctly for each end user.

Create the required administrative users with the correct filters.

1. Create an administrative account at both the data source and the data target.

   See "Setting the User Name and Password" on page 272.

   Analytic Services uses this account to log onto the data source to retrieve data and to perform outline synchronization operations.

2. Create read and write filters to determine what administrators can view and update.

   See Chapter 36, "Managing Security for Users and Applications".

   - For replicated partitions, set up read filters at the data source to determine which data Analytic Services reads when replicating and set up write filters at the data target to determine which data Analytic Services writes to when replicating.
   - For transparent partitions, set up read filters at the data source to determine which data Analytic Services retrieves for end users and set up write filters at the data source to determine which data Analytic Services updates for end users.

Case Studies for Designing Partitioned Databases

The following sections describe examples of partitioning a database:

Case Study 1: Partitioning an Existing Database

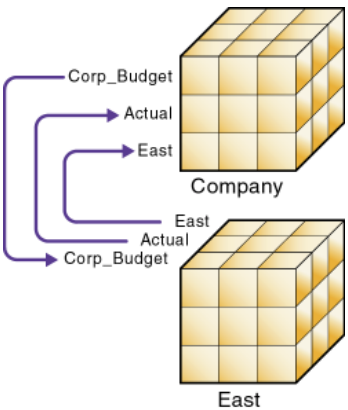
Assume that TBC, the fictional soft drink company upon which the Sample Basic database is based, started out with a centralized database. As the eastern region grew, however, this solution was no longer feasible. The networks to the eastern region could not handle the large flow of data. Users were constantly waiting for data that they needed to make decisions. One day, the network went down and users at the eastern region could not access the data at all.

Everyone agreed that the eastern region needed to access its own data directly, without going through the company database. In addition, TBC decided to change where budgeting information was stored. The corporate budget stays at company headquarters, but the eastern region budget moves to the eastern region's database.

So, assume that TBC decided to ask you to partition their large centralized database into two smaller databases—Company and East.

This example is based on the Samppart application, which contains the Company database, and the Sampeast application, which contains the East database. Both are shipped with Analytic Services.

This illustration shows a subset of the partitioned databases. The arrows indicate flow from the data source to the data target. The Company database is the data source for the Corp_Budget member and the data target for the East and the East Actual members. The East database is the data source for its East and Actual members and the data target for the Corp_Budget member.



➤ Use this procedure to create a partition based on this example:

1. Determine which data to partition.

   The Sample Basic database contains five standard dimensions—Year, Measures, Product, Market, and Scenario.

   - Partition the database along the East member of the Market dimension to give the eastern region more control over the contents of its database.
   - Partition the database along the Actual and Corp_Budget members of the Scenario dimension.

2. Choose the data source and the data target.
   - For Corp_Budget, use Company as source and East as Target; because the company owns the corporate budget, it is the source.
   - For Eastern Region and Actual, East is the source and Company is the target, because the eastern region needs to update its market and actual information.
3. Decide on the type of partition to use.
   - For East, use transparent because the data target (Company) needs up-to-the-minute data.
   - For Corp_Budget, use transparent because the data target (East) needs up-to-the minute data.
   - For East Actual, use replication because the data target (Company) does not need up-to-the-minute data.
4. Finally, create the partitioned databases by performing the following tasks.
   - Creating the new Sampeast application.
   - Creating the new East database by cutting the Company outline and pasting it into the East outline. Then delete the extra members (that is, South, West, and Central) and promote East.
   - If necessary, editing existing data sources, rules files, calculation scripts, report scripts, and outlines.
   - Creating the partitions.
   - Loading data into the new partitions.

Now that the corporate database is partitioned, users and database administrators see the following benefits:

- Faster response times, because they are competing with fewer users for the data and they are accessing the data locally.
- Database administrators can control the down time of their local databases, making them easier to maintain.
- Access to more data—now users can connect to both the eastern and corporate budgets.
- Higher quality data, because the corporate budget and eastern budget are now synchronized, they use the same data.

Case Study 2: Connecting Existing Related Databases

Assume that TBC has several databases, such as Inventory, Payroll, Marketing, and Sales. Users viewing the Sample Basic database want to share data with and navigate to those other databases and you, the database administrator, want to synchronize related data. It is impractical to combine all of the databases into one database, for the following reasons:

- So many users access it that performance is slow.
- You cannot find a down time to administer the database.
- No one has control over their own data, because it is centrally managed.
- The database is very sparse, because so much of the data is unrelated.

By connecting the databases instead, you can reap the following benefits:

- Leverage work that has already been completed.
- Synchronize the data.

So you decide to connect multiple databases.

Note: This example is not shipped with Analytic Services.

1. Determine which data to connect. First, connect the Inventory database.
   - Replicate the Opening_Inventory and Ending_Inventory members from the Measures dimension of the Inventory database into the Measures dimension of the Sample Basic database.
   - Do not replicate the Number_On_Hand, Number_Shipped, and Number_Returned members in the Measures dimension of the Inventory database to the Sample Basic database.
   - Add a link to the Inventory database so that users can view these more detailed measures if they need to.
   - Create a partition containing data from the Payroll, Marketing, and Sales databases in the Sample Basic database.
2. Choose the data source and the data target. In the case of the Opening_Inventory and Ending_Inventory members, the Inventory database is the data source and the Sample Basic database is the data target.

3. Decide on the type of partition to use.

   Use a replicated partition for the Opening_Inventory and Ending_Inventory members because the network connection is slow.

4. Connect the Payroll, Marketing, and Sales databases. Perform the tasks in step 1 through step 3 for each database.

5. Finally, create the partitioned databases by performing the following tasks:

   - Editing existing data sources, rules files, calculation scripts, report scripts, and outlines
   - Creating the partitions
   - If necessary, loading data into the new partitions

Now that the Sample Basic database is partitioned, users and database administrators see the following benefits:

- Database administrators can control the down time of their local databases, making them easier to maintain.
- Access to more data—now users can link to new databases.
- Higher quality data, because the databases are now synchronized, that is, they use the same data.

Case Study 3: Linking Two Databases

Assume that TBC, the fictional soft drink company upon which the Sample Basic database is based, has two main databases—the Sample Basic database and TBC Demo. Both databases have similar outlines, but TBC Demo has two additional dimensions, Channel, which describes where a product is sold, and Package, which describes how the product is packaged.

The database administrator for the Sample Basic database notices that more and more users are requesting that she add channel information to the Sample Basic database. But, since she does not own the data for channel information, she is reluctant to do so. She decides instead to allow her users to link to the TBC Demo database which already contains this information.

Note: This example is not shipped with Analytic Services.

Here are the steps to take:

1. Determine which data to link.

   The database administrator decides to link the Product dimension of the Sample Basic database to the Product dimension of TBC Demo. Users can then drill across to TBC Demo and view the Channel and Package information.

2. Choose the data source and the data target. Because users start at the Sample Basic database, it is considered the data target. Likewise, because users move to TBC Demo, it is considered the data source.

   Note: This setup is the opposite of replicated and transparent databases, where users move from the data target to the data source.

3. Decide on the type of partition to use.

   Use a linked partition because the databases have different dimensionality.

4. Finally, create the partition:

   - Establish a link from the Product member of the Sample Basic database to the Product dimension of the TBC Demo database. Remember to map the extra dimensions from TBC Demo, Channel and Product, to `void` in the Sample Basic database. For more information, see "Mapping Data Cubes with Extra Dimensions" on page 275.
   - Set up a guest account on TBC Demo that gives the users who connect from the Sample Basic database permissions to access the Channel and Package dimensions. For a general discussion on creating accounts, see "Granting Permissions to Users and Groups" on page 840. For directions on how to assign accounts to linked partitions, see "Choosing a Partition Type" on page 271 and "Choosing a Partition Type" on page 271.

Now that the databases are linked, users and database administrators see the following benefits:

- Users have access to more data than before.
- The database administrator for the Sample Basic database does not need to maintain the TBC Demo database, all she needs to do is check the link periodically to make sure that it still works.

Chapter 14

Creating and Maintaining Partitions

When you build a new partition, each database in the partition uses a partition definition file to record all information about the partition, such as its data source and data target and the areas to share. You must have Database Designer permissions or higher to create a partition.

Note: The information in this chapter is designed for block storage databases. Some of the information is not relevant to aggregate storage databases. For detailed information on the differences between aggregate and block storage, see Chapter 57, "Comparison of Aggregate and Block Storage." For information on creating aggregate storage applications, see Chapter 58, "Aggregate Storage Applications, Databases, and Outlines."

This chapter contains the following sections that describe how to create a replicated, transparent, or linked partition:

CAUTION: You must design partitions carefully. Hyperion strongly recommends that you read Chapter 13, "Designing Partitioned Applications," before creating partitions.

After you create a partition, you must maintain the partition. This chapter contains the following sections that describe how to maintain an existing partition:

Process for Creating Partitions

Here is the suggested process for creating database partitions.

1. Set the partition type as replicated, transparent, or linked. See "Choosing a Partition Type" on page 271.
2. Set up the data source and the data target, including specifying the location of the data source, the location of the data target, notes to describe each, and the source outline. See "Setting up the Data Source and the Data Target" on page 271.
3. Set up the administrative account to use to connect the data source and the data target partitions. See "Setting the User Name and Password" on page 272.
4. Define the area of the database to partition. See "Defining a Partition Area" on page 273.
5. If necessary, map the members in the data source to the members in the data target. See "Mapping Members" on page 273.
6. Validate the partition. See "Validating Partitions" on page 281.
7. Save the partition. See "Saving Partitions" on page 283.
8. If the partition is replicated, populate the partition. See "Populating or Updating Replicated Partitions" on page 289.

9. Load and calculate the new database that contains the partition. Loading and calculating the partition may require you to change existing rules files and calculation scripts. See Chapter 16, "Understanding Data Loading and Dimension Building" and Chapter 21, "Calculating Analytic Services Databases."

Choosing a Partition Type

When you create a partition, choose one of the following types:

- A replicated partition is a copy of a portion of the data source that is stored in the data target. For detailed information, see "Replicated Partitions" on page 242.
- A transparent partition allow users to access data from the data source as though it were stored in the data target. However, the data is stored at the data source, which can be in another application or in another Analytic Services database or on another Analytic Server. For detailed information, see "Transparent Partitions" on page 248.
- A linked partition sends users from a cell in one database to a cell in another database. A linked partition gives users a different perspective on the data. For detailed information, see "Linked Partitions" on page 256.

➤ To choose a partition type, see "Specifying the Partition Type and Settings" in the *Essbase Administration Services Online Help*.

Setting up the Data Source and the Data Target

You must set up the data source and the data target, including specifying their location, entering notes about each one (optional), and specifying the source outline.

➤ To set up the data source and the data target:

1. Specify the location of the data source.

   Specify the names of the Analytic Server, application, and database to use as the data source. See "Specifying Connection Information for Partitions" in the *Essbase Administration Services Online Help*.

2. Specify the location of the data target.

   Specify the names of the Analytic Server, application, and database to use as the data target. See "Specifying Connection Information for Partitions" in the *Essbase Administration Services Online Help*.

   Note: Do not use network aliases, such as localhost, for the data source or data target names unless you are *certain* that they are propagated to all computers on your system. If you're not certain, use the full server name. This is especially important for linked partitions, because this is the host name that clients connected to the data target use to find the data source.

3. If desired, enter a note to describe the data source or data target.

   See "Specifying Connection Information for Partitions" in the *Essbase Administration Services Online Help*.

4. If desired, specify the outline to which you can make changes.

   By default, all changes made on the data source outline overwrite the data target outline when you synchronize the outlines. You can, however, specify that changes made to the data target outline overwrite the data source outline when you synchronize the outlines. For more information, see "Synchronizing Outlines" on page 284.

Setting the User Name and Password

You must specify a user name and password for Analytic Services to use when communicating between the data source and the data target. The user name and password must be identical on both the data source and the data target. Analytic Services uses this user name and password to:

- Transfer data between the data source and the data target for replicated and transparent partitions. Local security filters apply to prevent end users from seeing privileged data.
- Synchronize database outlines for all partition types.

For more information, see "Planning for Security for Partitioned Databases" on page 261.

➤ To set the user name and password for the data source and the data target, see "Specifying Connection Information for Partitions" in *Essbase Administration Services Online Help*.

Defining a Partition Area

You can define or edit the areas of the data source to share with the data target in a partition. An *area* is a subcube within a database. For example, an area could be all Measures at the lowest level for Actual data in the Eastern region. A partition is composed of one or more areas.

When you define a replicated area, make sure that both the data source and data target contain the same number of cells. This verifies that the two partitions have the same shape. For example, if the area covers 18 cells in the data source, the data target should contain an area covering 18 cells into which to put those values. The cell count does not include the cells of attribute dimensions.

For more information on partition areas, see "Determining Which Data to Partition" on page 241.

Note: Use member names instead of their aliases to create area definitions. Although Analytic Services validates the aliases, the partitions will not work.

➤ To define a partition area, see "Defining Areas in Partitions" in *Essbase Administration Services Online Help*.

Mapping Members

To create a partition, Analytic Services must be able to map all shared data source members to data target members. Hyperion recommends that data source member names and data target member names are the same to reduce the maintenance requirements for the partition, especially when the partition is based on member attributes.

If the data source and the data target contain the same number of members and use the same member names, Analytic Services automatically maps the members. You need only validate, save, and test the partitions, as described in "Validating Partitions" on page 281, "Saving Partitions" on page 283, and "Testing Partitions" on page 283. If Analytic Services cannot map automatically, you must map manually.

Map data source members to data target members in any of the following ways:

- Enter or select member names in manually.
- Import the member mappings from an external data file.
- Create area-specific mappings.

To map members, see “Defining Global Mappings in Partitions” in *Essbase Administration Services Online Help*.

The following sections provide details about mapping members:

Mapping Members with Different Names

If the data source outline and data target outline contain different members or if the members have different names in each outline, you must map the data source members to the data target members. In the following example, the first two member names are identical, but the third member name is different:

Source	Target
Product	Product
Cola	Cola
Year	Year
1998	1998
Market	Market
East	East_Region

Because you know that East in the data source corresponds to East_Region in the data target, map East to East_Region. Then, all references to East_Region in the data target point to East in the data source. For example, if the data value for Cola, 1998, East is 15 in the data source, the data value for Cola, 1998, East_Region is 15 in the data target.

Mapping Data Cubes with Extra Dimensions

The number of dimensions in the data source and data target may vary. The following example illustrates a case where there are more dimensions in the data source outline than in the data target outline:

Source	Target
Product	Product
Cola	Cola
Market	Market
East	East
Year	
1999	
1998	
1997	

If you want to map member 1997 of the Year dimension from the data source to the data target, you can map it to Void in the data target. But first, you must define the areas of the data source to share with the data target:

Source	Target
@DESCENDANTS(Market), 1997	@DESCENDANTS(Market)

You can then map the data source member to Void in the data target:

Source	Target
1997	Void

"Void" is displayed automatically; entering "Void" yourself may cause errors.

If you do not include at least one member from the extra dimension in the area definition, you will receive an error message when you attempt to validate the partition.

Note: When you map a member from an extra dimension, the partition results reflect data only for the mapped member. In the above example, the Year dimension contains three members: 1999, 1998, and 1997. If you map member 1997 from the data source to the data target, then the partition results reflect Product and Market data only for 1997. Product and Market data for 1998 and 1999 will not be extracted.

The following example illustrates a case where the data target includes more dimensions than the data source:

Source	Target
Product	Product
Cola	Cola
	Market
	East
Year	Year
1997	1997

In such cases, you must first define the shared areas of the data source and the data target:

Source	Target
@IDESCENDANTS(Product)	@IDESCENDANTS(Product), East

You can then map member East from the Market dimension of the data target to Void in the data source:

Source	Target
Void	East

If member East from the Market dimension in the data target is not included in the target areas definition, you will receive an error message when you attempt to validate the partition.

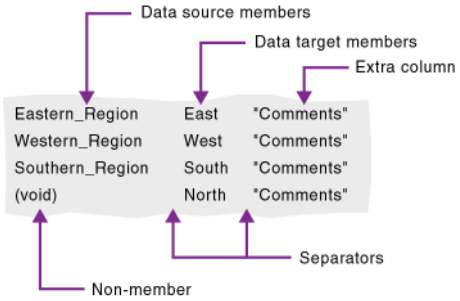
Mapping Shared Members

When you create a replicated or transparent partition using a shared member, use the real member names in the mapping. Analytic Services maps the real member, not the shared one, from the data source.

Importing Member Mappings

You can import member mappings from a text file. Mapping files must end in `.txt`. A sample member file must contain all of the following (except extra columns):

Figure 80: Member Mapping Import File



- Data target members column—lists the member names in the data target. Member names containing spaces must be in quotes.
- Data source members column—lists the member names in the data source. Member names containing spaces must be in quotes.
- Non-member column—missing members. Use when you are mapping an extra member in the data source to Void in the data target or vice versa.
- Separators—separate the columns. Separators can be tabs or spaces.
- Extra column—the file can contain extra columns that do not contain member names.

➤ To import member mappings, see "Importing Member Mappings for Partitions" in *Essbase Administration Services Online Help*.

Mapping Attributes Associated with Members

You must accurately map attribute dimensions and members from the data source to the data target to ensure that the partition is valid.

Note: You cannot map members of attributes dimension in replicated partitions. For more information, refer to "Rules for Replicated Partitions" on page 243. You can, however, map attributes in transparent and linked partitions. For information on using attributes in partitions, see "Attributes in Partitions" on page 238.

In the following example, the outline for the data source contains a Product dimension with a member 100 (Cola). Children 100-10 and 100-20 are associated with member TRUE of the Caffeinated attribute dimension, and child 100-30 is associated with member FALSE of the Caffeinated attribute dimension.

The data target outline has a Product dimension with a member 200 (Cola). Children 200-10 and 200-20 are associated with member Yes of the With_Caffeine attribute dimension, and child 200-30 is associated with No of the With_Caffeine attribute dimension.

First define the areas to be shared from the data source to the data target:

Source	Target
`@DESCENDANTS(100)`	`@DESCENDANTS(200)`
`@DESCENDANTS(East)`	`@DESCENDANTS(East)`

Then map attributes as follows:

Source	Target
`100-10`	`200-10`
`100-20`	`200-20`
`100-30`	`200-30`
`Caffeinated`	`With Caffeine`
`Caffeinated_True`	`With_Caffeine_True`
`Caffeinated_False`	`With_Caffeine_No`

If you map attribute Caffeinated_True to attribute With_Caffeine_No, you receive an error message during validation. You must associate caffeinated cola from the data source to caffeinated cola in the data target.

There can be instances where an attribute dimension or an attribute member exists in the outline of the data source but not in the outline of the data target, or vice versa. For example:

Source	Target

`Caffeinated`
`  True`
`  False`	

In such cases, you have the following choices:

- Create the Caffeinated attribute dimension and its members in the outline of the data target and associate them with the Product dimension. You can then map the attributes from the data source to the data target.
- Map the Caffeinated attribute dimension in the data source to Void in the data target.

For a comprehensive discussion of attributes, see Chapter 10, "Working with Attributes." For a general discussion of attributes in partitions, see "Attributes in Partitions" on page 238.

Creating Advanced Area-Specific Mappings

If you can map all of the members in your data source to their counterparts in the data target using standard member mapping, then you don't need to perform advanced area-specific mapping.

If, however, you need to control how Analytic Services maps members at a more granular level, you may need to use area-specific mapping. Area-specific mapping maps members in one area to members in another area only in the context of a particular area map.

Use area-to-area mapping when you want to:

- Map data differently depending on where it is coming from.
- Map more than one member in the data source to a single member in the data target.

Since Analytic Services cannot determine how to map multiple members in the data source to a single member in the data target, you must logically determine how to divide your data until you can apply one mapping rule to that subset of the data. Then use that rule in the context of area-specific mapping to map the members.

➤ To create area-specific mappings, see "Defining Area-Specific Member Mappings in Partitions (Optional)" in *Essbase Administration Services Online Help*.

Example 1: Advanced Area-Specific Mapping

The data source and data target contain the following dimensions and members:

Source	Target
Product	Product
Cola	Cola
Market	Market
East	East
Year	Year
1998	1998
1999	1999
	Scenario
	Actual
	Budget

The data source does not have a Scenario dimension. Instead, it assumes that past data is actual data and future data is forecast, or budget, data.

You know that 1998 in the data source should correspond to 1998, Actual in the data target and 1999 in the data source should correspond to 1999, Budget in the data target. So, for example, if the data value for Cola, East, 1998 in the data source is 15, then the data value for Cola, East, 1998, Actual in the data target should be 15.

Because mapping works on members, not member combinations, you cannot simply map 1998 to 1998, Actual. You must define the area (1998 and 1998, Actual) and then create area-specific mapping rules for that area.

Because the data source does not have Actual and Budget members, you must also map these members to Void in the data target.

Example 2: Advanced Area-Specific Mapping

You can also use advanced area-specific mapping if the data source and data target are structured very differently but contain the same kind of information.

This strategy works, for example, if your data source and data target contain the following dimensions and members:

Source	Target
Market	Customer_Planning
NY	NY_Actual
CA	NY_Budget
	CA_Actual
	CA_Budget
Scenario	
Actual	
Budget	

You know that NY and Actual in the data source should correspond to NY_Actual in the data target and NY and Budget in the data source should correspond to NY_Budget in the data target. So, for example, if the data value for NY, Budget in the data source is 28, then the data value for NY_Budget in the data target should be 28.

Because mapping works on members, not member combinations, you cannot simply map NY, Actual to NY_Actual. You must define the area (NY and Actual, and NY_Actual) and then create area-specific mapping rules for that area.

Because the data target does not have NY and CA members, you must also map these members to Void in the data target so that the dimensionality is complete when going from the data source to the data target.

Validating Partitions

When you create a partition, validate it to ensure that it is accurate before you use it. In order to validate a partition, you must have Database Designer permissions or higher. After you validate, save the partition definition. If necessary, you can edit an existing partition.

When Analytic Services validates a partition definition, it checks on the Analytic Server for the data source and the data target to ensure that:

- The area definition is valid (contains no syntax errors).
- The specified data source members are valid and map to valid members in the data target.

- All connection information is correct; that is, the server names, database names, application names, user names, and password information.
- For linked partitions, the default user name and password that you provide are correct.
- For replicated and transparent partitions, a replication target does not overlap with a replication target; a replication target does not overlap with a transparent target; a transparent target does not overlap with a transparent target; and a replication target does not overlap with a transparent target.
- For replicated and transparent partitions, the cell count for the partition is the same on the data source and the data target.
- For replicated and transparent partitions, the area dimensionality matches the data source and the data target.
- You must validate a transparent partition that is based on attribute values to ensure that the results are complete. Analytic Services does not display an error message when results are incomplete.

After you validate, save the partition. When you save a partition, the partition definition is saved to two different `.ddb` files, on both the data source server and the data target server.

➤ To validate a partition, use any of the following methods:

Tool	Topic	Location
Administration Services	Validating Partitions	*Essbase Administration Services Online Help*
ESSCMD	VALIDATEPARTITIONDEFFILE	*Technical Reference*
MaxL	**create partition**	*Technical Reference*

Saving Partitions

After you validate the partition definition, you can save the partition definition to any of the following locations:

- To both the data source server and the data target server. The partition definition is stored in two `.ddb` files.
- To a client machine. The partition definition is stored in a single `.ddb` file.

To save a partition definition, see "Saving Partitions" in the *Essbase Administration Services Online Help*.

Process for Maintaining Partitions

Here is the suggested process for maintaining database partitions.

1. Test the partition. See "Testing Partitions" on page 283.
2. Synchronize the outlines of the data target and the data source. See "Synchronizing Outlines" on page 284.
3. Update or populate replicated partitions. See "Populating or Updating Replicated Partitions" on page 289.
4. Edit or delete existing partitions. See "Editing and Deleting Partitions" on page 290.
5. Viewing information about existing partitions. See "Viewing Partition Information" on page 291.
6. Troubleshoot partitions. See "Troubleshooting Partitions" on page 291.

Testing Partitions

To test a partition:

- View data targets using the Spreadsheet Add-in or other tool to make sure that the user sees the correct data.
- When testing a linked partition, make sure that Analytic Services links you to the expected database and that the default user name and password work correctly.

Synchronizing Outlines

When you partition a database, Analytic Services must be able to map each dimension and member in the data source outline to the appropriate dimension and member in the data target outline. After you map the two outlines to each other, Analytic Services can make the data in the data source available from the data target as long as the outlines are synchronized and the partition definitions are up-to-date.

If you make changes to one of the outlines, the two outlines are no longer synchronized. Although Analytic Services does try to make whatever changes it can to replicated and transparent partitions when the outlines are not synchronized, Analytic Services may not be able to make the data in the data source available in the data target.

However, Analytic Services tracks changes that you make to your outlines and provides tools to make it easy to keep your outlines synchronized.

This section describes Analytic Services synchronizes outlines.

- "Setting the Source Outline and the Target Outline" on page 284
- "Performing Outline Synchronization" on page 286
- "Tracking Changes" on page 286
- "Updating Shared Members During Outline Synchronization" on page 287

Setting the Source Outline and the Target Outline

Before you can synchronize your outlines, you must determine which outline is the source outline and which is the target outline.

- The *source outline* is the outline that outline changes are taken from.
- The *target outline* is the outline that outline changes are applied to.

By default, the source outline is from the same database as the data source; that is, outline and data changes flow in the same direction. For example, if the East database is the data source and the Company database is the data target, then the default source outline is East.

You can also use the data target outline as the source outline. You might want to do this if the structure of the outline (its dimensions, members, and properties) is maintained centrally at a corporate level, while the data values in the outline are maintained at the regional level (for example, East). This allows the database administrator to make changes in the Company outline and apply those changes to each regional outline when she synchronizes the outline.

If you make changes to the:

- Shared area in the source outline, you can propagate these changes to the target outline when you synchronize the outlines.
- Target outline, those changes cannot be propagated back to the source outline when you synchronize the outlines. To move these changes up to the source outline, make those changes in Outline Editor. See Chapter 8, "Creating and Changing Database Outlines."

Analytic Services updates as many changes as possible to the target outline. If Analytic Services cannot apply all changes, a warning message prompts you to see the application log for details. Messages that pertain to outline synchronization are prefixed with OUTLINE SYNC. For more information, see in "Viewing the Analytic Server and Application Logs" on page 997.

➤ To set the source outline, see "Setting up the Data Source and the Data Target" on page 271.

Performing Outline Synchronization

➤ To synchronize outlines, use any of the following methods:

Tool	Topic	Location
Administration Services	Synchronizing Outlines	*Essbase Administration Services Online Help*
MaxL	**refresh outline**	*Technical Reference*
ESSCMD	GETPARTITIONOTLCHANGES APPLYOTLCHANGEFILE RESETOTLCHANGETIME PURGEOTLCHANGEFILE	*Technical Reference*

Note: For synchronizing non-Unicode-mode outlines with multi-byte characters, you can use only non-Unicode clients such as ESSCMD or MaxL statements executed through the MaxL Shell.

Note: Outline synchronization cannot be performed on an outline containing a Dynamic Calc member that has many (approximately 100 or more) children.

Tracking Changes

The following table describes what happens when you change the source outline and then synchronize the target outline with the source outline:

Action You Take	Action Analytic Services Takes
Make changes to the source outline	1. Records the changes in a change log named `ess`*`xxxxx`*`.chg`, where *xxxxx* is the number of the partition. If you have more than one partition on a source outline, Analytic Services creates a *change log* for each partition. 2. Creates or updates the outline change timestamp for that partition in the `.ddb` file. Each partition defined against the source outline has a separate timestamp in the `.ddb` file.

Action You Take	Action Analytic Services Takes
Pull changes from the outline source	1. Compares the last updated timestamp in the target outline's `.ddb` file to the last updated timestamp in the source outline's `.ddb` file. Analytic Services updates the target timestamp when it finishes synchronizing the outlines using the last updated time on the *source outline*, even if the two outlines are on servers in different time zones. 2. If the source outline has changed since the last synchronization, Analytic Services retrieves those changes from the source outline's change log and places them in the target outline's change log. The change logs may have different names on the source outline and the target outline.
Select the changes to apply to the target outline	1. Applies the changes to the target outline. 2. Updates the timestamp in the target outline's `.ddb` file, using the time from the source outline.

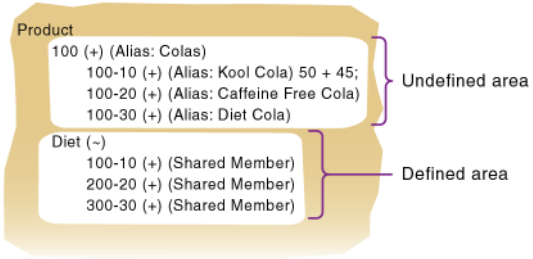
CAUTION: If you choose *not* to apply some changes, you cannot apply those changes later.

Updating Shared Members During Outline Synchronization

An actual member and its shared members in the source outline are propagated to the target outline if at least one actual or shared member is defined in the partition area. In illustrated in Figure 81, the partition definition is @IDESC("Diet"). The parent 100 and its children (100-10, 100-20, 100-30) are not defined in the

partition area. The parent Diet and its children (100-10, 100-20, 100-30) are defined in the partition area. The children of Diet are shared members of the actual members.

Figure 81: Shared Members and Outline Synchronization



If you make a change to an actual member in the undefined partition area, such as adding an alias to the 100-10 actual member, that change is propagated to the target outline because it is associated with a shared member in the defined partition area.

The reverse is also true. If a shared member is not in the partition area and its actual member is, a change to the shared member in the undefined area is propagated to the target outline.

Any change made to a member that does not have at least one actual member (or shared member) in the defined partition area is not propagated to the target outline. For example, in Figure 81, a change to the parent 100 is not propagated to the target outline because it is in the undefined partition area and does not have an associated shared member in the defined partition area.

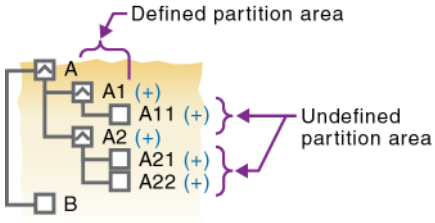
If a shared member is included in the partition area, then it is recommended to include its parent. In the above example, the parent Diet is included in the outline because its children are shared members and in the defined partition area.

Implied shared members are treated the same as shared members during outline synchronization. Actual members and their implied shared members in the source outline are propagated to the target outline if at least one actual or implied shared member is defined in the partition definition.

Using the partition definition as @CHILD("A") in the example in Figure 82, A1 and A2 are in the defined partition area, and A11, A21, A22 are in the undefined partition area. Although A11 (implied shared member) is in the undefined partition area, a change to A11 is propagated to the target outline because its parent, A1, is in the defined partition area. The change to the children A21 and A22 is not

propagated to the target outline because these members are not defined in the partition area and are not associated with a member that is in the defined partition area.

Figure 82: Implied Shared Members and Outline Synchronization



The reverse is true again. If A1 is not defined in the partition area and its implied shared member is, then any change to A1 is propagated to the target outline.

Populating or Updating Replicated Partitions

The database administrator should regularly update data in a replicated partition. How frequently you update replicated partitions depends on users' requirements for up-to-the-minute data. Analytic Services keeps track of when the data source was last changed and when the data target was last updated so that you can determine when to update replicated partitions. This information is saved at the data source. Either database administrator—that of the data source site or that of the data target site—can be responsible for replicating data.

Analytic Services also tracks which cells in a partition are changed. You can choose to update:

- Just the cells that have changed since the last replication—It is fastest to update just the cells that have changed.

  Note: If you've deleted data blocks on the data source, Analytic Services updates all data cells at the data target even if you choose to update only changed cells. You can delete data blocks at the data source by using the CLEARDATA command in a calculation script; using "Clear combinations" in your rules file during a data load; issuing CLEAR UPPER, CLEAR INPUT or RESETDB commands in Administration Services; restructuring the database keeping only level 0 or input data; or deleting sparse members.

- All cells—This is much slower. You may need to update all cells if you are recovering from a disaster where the data in the data target has been destroyed or corrupted.

You can replicate:

- All data targets connected to a data source. For example, if you replicate all data targets connected to the Sampeast East database, Analytic Services updates the Budget, Actual, Variance, and Variance % members in the Samppart Company database:
- From all data sources connected to a data target. For example, if you replicate from all data sources connected to the Samppart Company database, Analytic Services pulls the Budget, Actual, Variance, and Variance % members from the Sampeast East database and updates them in the Samppart Company database.

➤ To update a replicated partition, use any of the following methods:

Tool	Topic	Location
Administration Services	Replicating Data	*Essbase Administration Services Online Help*
MaxL	**refresh replicated partitio**n	*Technical Reference*
ESSCMD	GETUPDATEDREPLCELLS GETALLREPLCELLS PUTUPDATEDREPLCELLS PUTALLREPLCELLS	*Technical Reference*

Editing and Deleting Partitions

You can edit and delete existing partitions. When you edit a partition, you use the same interface that you used to create the partition, making modifications as necessary.

When you delete a partition, Analytic Services deletes the partition definition from the `.ddb` file on the data source and data target servers.

➤ To edit or delete a partition, see “Opening the Create or Edit Partition Window” and “Deleting Partitions” in *Essbase Administration Services Online Help*.

Viewing Partition Information

➤ To view information about a partition, use any of the following methods:

Tool	Topic	Location
Administration Services	Opening the Create or Edit Partition Window	*Essbase Administration Services Online Help*
MaxL	**display partition**	*Technical Reference*
ESSCMD	PRINTPARTITIONDEFFILE	*Technical Reference*

Troubleshooting Partitions

The following table lists common problems that you may encounter when using partitions.

Symptom	Possible Causes	Solutions
When replicating to multiple data targets, some are not replicated.	The connection between the data source and one of the data targets was lost during the replication operation.	Retry the replication operation. If one database is unavailable, replicate into just the databases that are available.
Not all information arrived at the data target.	The data source and the data target outlines are no longer mappable.	Synchronize outlines for the data source and the data target and try again.
A new or recently changed partition is validated and saved but does not function.	The partition may have a circular dependency. If database A is the source for database B, then database B cannot be source for database A for the same slice. Such invalid partitions can be saved to the server MaxL, but they are disabled at the server.	Edit the partion definition to remove the circular dependency.
You keep running out of ports.	Partitions connect to other databases using ports.	Purchase more ports.

Symptom	Possible Causes	Solutions
When you try to access a partition, you cannot connect to it.	Someone has deleted, renamed, or moved the application containing the database to which you are trying to connect.	Edit the partition having problems and specify the new application name or location.
Partitioned databases can no longer connect to each other.	Your host names may not match. Did you use the `hosts` file to provide aliases to your local machine?	Make sure that the host names are synchronized between the servers.
Analytic Services overwrites user edits.	Users are changing data at a replicated partition that you overwrite each time that you update the partition.	Set the partition to not allow user updates or explain to users why their data disappears.
Administration Services does not reflect outline changes; that is, it lists the outlines as being in sync even though one outline has changed.	Was the target outline changed, but not the source outline? Analytic Services only propagates changes to the source outline. Does the outline change affect a defined partition? Analytic Services does not propagate changes to the outline that do not affect any partitions.	Examine the partition definition.
Data is confusing.	Your partition may not be set up correctly.	Check your partition to make sure that you are partitioning the data that you need.
Moved members or dimensions in the source outline are not reflected properly in a synchronized target outline.	Moving a dimension past a dimension that is not in the partition may not get propagated to the target during outline synchronization. Also, moving a member past a member that is not in the partition may not get propagated.	If possible, structure your outline so that members and dimensions are not moved across partitions. If this is not possible, change the target outline to reflect the source outline moved members or dimensions.

Chapter 15

Accessing Relational Data with Hybrid Analysis

Because relational databases can store several terabytes of data, they offer nearly unlimited scalability. Multidimensional databases are generally smaller than relational databases but offer sophisticated analytic capabilities. With Essbase Hybrid Analysis, you can integrate a relational database with an Essbase Analytic Services database and thereby leverage the scalability of the relational database with the conceptual power of the multidimensional database.

Hybrid Analysis eliminates the need to load and store lower–level members and their data within the Analytic Services database. This feature gives Analytic Services the ability to operate with almost no practical limitation on outline size and provides for rapid transfer of data between Analytic Services databases and relational databases.

This chapter helps you understand Hybrid Analysis and explains how you can take advantage of its capabilities.

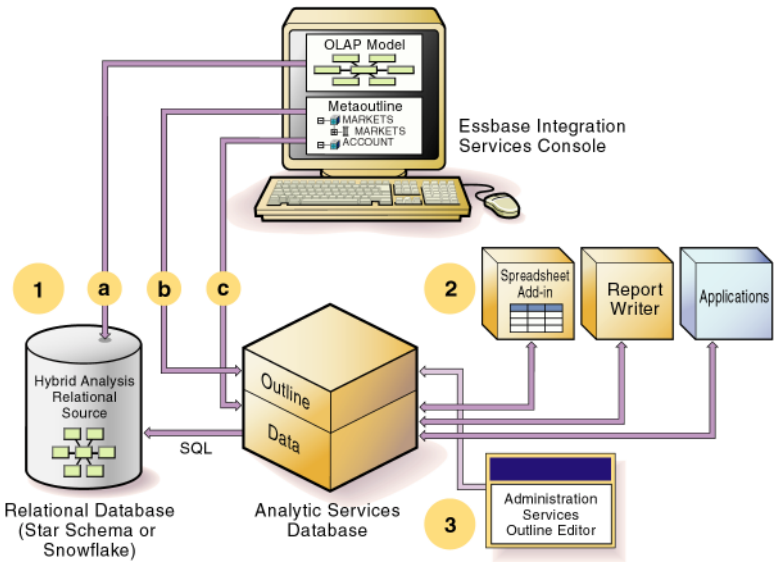
Note: The information in this chapter is designed for block storage databases. Some of the information is not relevant to aggregate storage databases. For detailed information on the differences between aggregate and block storage, see Chapter 57, "Comparison of Aggregate and Block Storage."

The chapter includes the following topics:

Understanding Hybrid Analysis

Hybrid Analysis integrates a relational database with an Analytic Services multidimensional database so that applications and reporting tools can directly retrieve data from both databases. Figure 83 illustrates the hybrid analysis architecture:

Figure 83: Hybrid Analysis Architecture



Hybrid Analysis Relational Source

The initial step in setting up Hybrid Analysis is to define the relational database as a hybrid analysis relational source (**1** in Figure 83).

You define the hybrid analysis relational source in Essbase Integration Services Console. (The individual tasks are discussed in "Defining Hybrid Analysis Relational Sources" on page 297.) Through Integration Services Console, you first specify the relational data source for the *OLAP model*. The OLAP model is a

schema that you create from tables and columns in the relational database. To build the model, Integration Services accesses the star schema of the relational database (**a** in Figure 83).

Using the model, you define hierarchies and tag levels whose members are to be enabled for hybrid analysis. You then build the *metaoutline*, a template containing the structure and rules for creating the Analytic Services outline, down to the desired hybrid analysis level. The information enabling hybrid analysis is stored in the *OLAP Metadata Catalog*, which describes the nature, source, location, and type of data in the hybrid analysis relational source.

Next, you perform a *member load* which adds dimensions and members to the Analytic Services outline (**b** in Figure 83). When the member load is complete, you run a data load to populate the Analytic Services database with data (**c** in Figure 83). At this point, the hybrid analysis architecture is in place:

- The lower level members and their associated data remain in the relational database.
- The data in the relational database is mapped to the Analytic Services outline that is defined by Hybrid Analysis.
- The outline resides in the Analytic Services database.

  Metadata is data that describes values within a database. The metadata that defines the hybrid analysis data resides in both the Analytic Services outline and in the Integration Services metaoutline on which the Analytic Services outline is based. Any changes that are made to hybrid analysis data in an OLAP model or metaoutline that is associated with an Analytic Services outline must be updated to the outline to ensure accuracy of the data reported in Analytic Services. See "Managing Data Consistency" on page 302 for information on keeping data and metadata in sync.
- Upper level members and their associated data reside in the Analytic Services database.

Data Retrieval

Applications and reporting tools, such as spreadsheets and Report Writer interfaces, can directly retrieve data from both databases (**2** in Figure 83). Using the dimension and member structure defined in the outline, Analytic Services determines the location of a member and then retrieves data from either the Analytic Services database or the hybrid analysis relational source. If the data

resides in the hybrid analysis relational source, Analytic Services retrieves the data through SQL commands. Data retrieval is discussed in "Retrieving Hybrid Analysis Data" on page 298.

If you want to modify the outline, you can use Outline Editor in Administration Services to enable or disable dimensions for hybrid analysis on an as–needed basis. (**3** in Figure 83). For information on using Outline Editor, see "Using Outline Editor with Hybrid Analysis" on page 301.

Hybrid Analysis Guidelines

Hybrid Analysis has some guidelines with which you should be familiar:

- A single Analytic Services database can be associated with only one hybrid analysis relational source.
- A hybrid analysis relational source can consist of only one relational database.
- A hybrid analysis enabled member should not be renamed. If a member is renamed, the member may not be retrieved the next time you perform a drill-through operation.
- Hybrid Analysis supports only parent–child prefixing on member names.
- Hybrid Analysis is not supported on accounts dimensions.
- Hybrid Analysis is not supported on user-defined dimensions.
- Only the lowest level members of a dimension can be enabled for hybrid analysis.
- Analytic Services does not support aliases for members that are enabled for hybrid analysis.
- Analytic Services requires the OLAP Metadata Catalog created in Integration Services in order to drill down in a hybrid analysis relational source.
- You can perform operations and analyses on dimensions that have attributes attached to one or more levels in an outline that is hybrid analysis enabled. The attribute dimension should be fully loaded into Analytic Services.
- Hybrid Analysis does not support scaling of measures dimension members using any of the operators + (addition), - (subtraction), * (multiplication), and / (division). If you use the scaling operators, drill-through queries into hybrid analysis data may show a mismatch between aggregated level-0 values in the Analytic Services database and the corresponding detail values in your relational data source.

- Analytic Services ignores all member properties, such as formulas, UDAs, and aliases for members that are enabled for hybrid analysis.
- Hybrid Analysis supports the Dynamic Times Series function.
- Hybrid Analysis does not support transparent, replicated, or linked partitions.
- Only the first hierarchy of a dimension with alternate hierarchies can have members enabled for hybrid analysis on its lowest levels.
- Hybrid Analysis does not support recursive hierarchies.
- Analytic Services supports drill-through operations defined on members that are enabled for Hybrid Analysis.
- When building a dimension that is enabled for hybrid analysis, you must ensure that the column in the relational table that contributes to the leaf level of the Analytic Services portion of the dimension is non-nullable.
- You can associate an attribute member with a member enabled for hybrid analysis.

Defining Hybrid Analysis Relational Sources

A hybrid analysis relational source is defined in Integration Services Console. Detailed information and the specific procedures for performing the following steps is available in Integration Services online help.

➤ To define a hybrid analysis relational source, perform the following steps:

1. Configure one or more relational data sources as described in the *Essbase Integration Services Installation Guide*.

   Note: If you are using two servers during hybrid analysis, the Data Source Name (DSN) must be configured on both servers and the DSN must be the same.

2. Create an OLAP model.

   The OLAP model star schema is created from tables and columns in the relational database. To build the model, Integration Services accesses the relational databases using the DSNs you configured in step 1.

3. In the OLAP model, define the hierarchies that you will use for hybrid analysis.

   Note: You can define any member of any dimension as enabled for hybrid analysis except members in an accounts dimension. This restriction is necessary because all members of an accounts dimension, including lower-level members, must remain in the Analytic Services database.

4. In the metaoutline, use the Build Multidimensional Down to Here command to select the level whose members will reside in the Analytic Services multidimensional database.
5. Use the Enable Hybrid Analysis Down to Here command to select the member levels that will remain in the relational database.
6. Run a member load to add dimensions and members to the outline.
7. Run a data load to populate the Analytic Services database with data.

Note: For detailed information on the above steps, see the Essbase Integration Services Console online help.

Retrieving Hybrid Analysis Data

In Hybrid Analysis, applications and reporting tools can directly retrieve data from both the relational and the Analytic Services databases by using the following tools:

- Essbase Spreadsheet Add-in
- Essbase Spreadsheet Services
- Report Writer
- Hyperion Analyzer
- Third–party applications

Note: The Analytic Services database and the relational database must be registered to the same ODBC data sources, and Integration Services must use the same source name for both databases.

Because data is being accessed from both the hybrid analysis relational source and the Analytic Services database when you perform calculations or generate reports, data retrieval time may increase with Hybrid Analysis; however, most capabilities of Analytic Services data retrieval operations are available with Hybrid Analysis, including pivot, drill-through, and other metadata-based methods.

Retrieving Hybrid Analysis Data with Spreadsheet Add-in

Use the Enable Hybrid Analysis option in the Essbase Options dialog box in Essbase Spreadsheet Add-in and Essbase Spreadsheet Services to drill down to members in the hybrid analysis relational source. Refer to the *Essbase Spreadsheet Add-in User's Guide* and to Spreadsheet Add-in online help for more information about these functions.

Supported Drill-Down Options in Hybrid Analysis

Hybrid Analysis supports the following drill–down options in Spreadsheet Add-in:

- Next Level (children)
- All Levels (all descendants)
- Bottom Level (level 0)
- All Siblings (all members with common parent)

For best performance, use children, bottom level, or siblings zoom-ins; avoid using a descendents zoom-in.

Supported Drill-Up Option in Hybrid Analysis

Hybrid Analysis supports the Parent drill-up option in Spreadsheet Add-in and Spreadsheet Services. However, the drill-up on a relational member always takes you to the leaf level member in the Analytic Services outline and not to the immediate parent of the relational member.

Retrieving Hybrid Analysis Data with Report Writer

In Report Writer, two commands, enable and disable respectively, Hybrid Analysis:

- <HYBRIDANALYSISON enables a report script to retrieve the members of a dimension that is enabled for hybrid analysis.
- <HYBRIDANALYSISOFF prevents a report script from retrieving the members of a dimension that is enabled for hybrid analysis.

The <ASYM and <SYM commands are not supported with Hybrid Analysis. If these commands are present in a report, errors may result. The <SPARSE command is ignored in reports retrieving data from a hybrid analysis relational source and does not generate errors.

The following is a sample Report Writer script that uses the IDESCENDANTS command to return hybrid analysis data:

```
<PAGE (Accounts, Scenario, Market)
Sales
Actual
<Column (Time)
<CHILDREN Time
<Row (Product)
<IDESCENDANTS 100-10
!
```

Retrieving Hybrid Analysis Data with Hyperion Analyzer

When you use Hyperion Analyzer, the procedures for retrieving hybrid analysis data are the same as the procedures for retrieving data that is not defined for Hybrid Analysis. (See the Hyperion Analyzer documentation for detailed information.)

For optimal performance when retrieving hybrid analysis data with Hyperion Analyzer, keep in mind the following guidelines:

- Always place dimensions that are enabled for Hybrid Analysis along the rows when constructing queries in Analyzer.
- Do not use any sort options if possible.

- Always use the children operator when drilling down to a hierarchy; do not use the descendants operator.
- Any additional processing in the form of restrictions or Top/Bottom retrievals may cause slower query times.

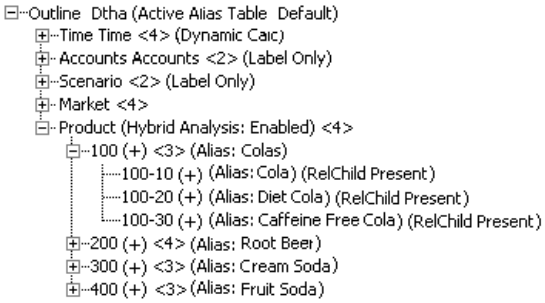
Using Outline Editor with Hybrid Analysis

In Outline Editor, you can toggle the Hybrid Analysis option button to enable or disable Hybrid Analysis for each dimension that is defined for hybrid analysis in Integration Services Console. If you open an outline that is not defined for hybrid analysis, the Hybrid Analysis option button is not displayed on the toolbar.

Note: When Hybrid Analysis is disabled for a dimension, the end user is unable to see and drill–through to the hybrid analysis data associated with the dimension; however, the members of the dimension are still visible in Outline Editor.

Figure 84 is an example of how an outline defined for hybrid analysis appears in Outline Editor. Note that dimensions that are enabled for hybrid analysis are identified to distinguish them from dimensions that are not enabled for hybrid analysis.

Figure 84: Example of Hybrid Analysis in Outline Editor



Managing Data Consistency

When you create a hybrid analysis relational source, the data and metadata are stored and managed in the relational database and in the Analytic Services database:

- Lower-level members and their associated data remain in the relational database.
- Data in the relational database is mapped to the Analytic Services outline which is defined for hybrid analysis.
- The outline resides in the Analytic Services database.
- Upper–level members and their associated data reside in the Analytic Services database.

Because data and metadata exist in different locations, information may become out of sync.

Analytic Services depends upon the OLAP Metadata Catalog in Integration Services to access the hybrid analysis relational source. At Analytic Services database startup time, Analytic Server checks the number of dimensions and members of the Analytic Services outline against the related metaoutline.

Any changes made to the associated OLAP model or metaoutline during an Integration Services session are not detected by the Analytic Server until the Analytic Services database is started again. Undetected changes can cause data inconsistency between the Analytic Services database and the hybrid analysis relational source.

If changes are made in the hybrid analysis relational source and members are added or deleted in the OLAP model or metaoutline, such changes can cause the Analytic Services outline to be out of sync with the metaoutline on which it is based. These types of changes and their effect on the hierarchical structure of a dimension are not reflected in the Analytic Services database until the outline build and data load process is completed through Integration Services Console.

In Essbase Administration Services, the Restructure Database dialog box has a check box that enables a warning whenever a restructuring affects an outline containing a hybrid analysis relational source. Such a problem occurs, for example, if members with relational children are moved or deleted.

Warnings are listed in the application log. You should decide if the warnings reflect a threat to data consistency. To view the application log, see "Viewing the Analytic Server and Application Logs" on page 997.

The Analytic Services administrator has the responsibility to ensure that the Analytic Services multidimensional database, the relational database, and the Integration Services OLAP model and metaoutline remain in sync. Both Administration Services and Integration Services Console provide commands that enable the administrator to perform consistency checks and make the appropriate updates.

For a comprehensive discussion of maintaining data consistency, see Chapter 46, "Ensuring Data Integrity."

For a comprehensive discussion of restructuring a database, see Chapter 52, "Optimizing Database Restructuring."

Managing Security in Hybrid Analysis

The Analytic Services administrator determines access to the hybrid analysis relational source on an individual Analytic Services user level. Access for Hybrid Analysis is governed by the same factors that affect overall Analytic Services security:

- Permissions and access levels for individual users and groups of users
- Permissions and access levels for the server, application, or database
- Specific database access levels for particular database members

If a security filter enables you to view only the relational children of the level 0 members that you have access to in Analytic Services, then you cannot view the relational children of the level 0 members that you do not have access to in Analytic Services.

Assume that you have the following outline, where San Francisco and San Jose are relational children of California, and Miami and Orlando are relational children of Florida:

```
Market
  West
    California
      San Francisco
      San Jose
  East
    Florida
      Miami
      Orlando
```

In this example, if a filter allows you to view only level 0 member California and its descendants, you can view California and its relational children, San Francisco and San Jose; however, you cannot view the children of level 0 member Florida.

For detailed information on Analytic Services security, see the following chapters:

- Chapter 36, “Managing Security for Users and Applications”
- Chapter 37, “Controlling Access to Database Cells”
- Chapter 38, “Security Examples”

Using Formulas with Hybrid Analysis

Formulas used with members enabled for hybrid analysis are subject to the following limitations:

- Formulas are supported only on a measures dimension.
- Formulas cannot be attached to relational members.
- Formulas cannot reference a relational member by name.
- Member set functions (such as @CHILDREN and @DESCENDANTS), which generate member lists in a formula, execute only in the Analytic Services portion of the outline.

If a formula or member enabled for hybrid analysis contains one or more functions that are not supported by Hybrid Analysis, Analytic Services returns the following error message:

```
Error executing formula for member
[member-name-to-which-formula-is-attached] (line [line#
where the offending function appears inside the formula):
function [Name of the offending function] cannot be used in
Hybrid Analysis.
```

Unsupported Functions in Hybrid Analysis

Hybrid Analysis does not support all Analytic Services functions. The following topics specify the categories of significant Analytic Services functions not supported by Hybrid Analysis.

Relationship Functions

Hybrid Analysis does not support functions that look up specific values in the database based on current cell location and a series of parameters. Some examples of these functions are given next.

@ANCEST
@SANCEST
@PARENT
@SPARENT
@CURLEV
@CURGEN

Member Conditions Functions

Hybrid Analysis does not support functions used to specify member conditions. Some examples of these functions are listed next.

@ISIANCEST
@ISIPARENT
@ISISIBLING
@ISMBR
@ISLEV
@ISSAMEGEN
@ISUDA

Range Functions

Hybrid Analysis does not support functions that use a range of members as arguments. Rather than return a single value, these functions calculate a series of values internally based on the range specified. Some examples of range functions that are not supported are listed next.

@PRIOR
@SHIFT
@PRIORS
@SHIFTS
@NEXT
@MDSHIFT
@MOVSUM
@MOVAVG
@ALLOCATE
@MDALLOCATE
@VAR
@VARPER
@MEDIAN
@RANK

Attribute Functions

Hybrid Analysis does not support any Analytic Services functions that deal with attributes. Some examples of these functions are listed next.

@ATTRIBUTEVAL
@ATTRIBUTESVAL
@WITHATTR

Current Member and XREF Functions

Hybrid Analysis does not support the following functions used to determine whether the current member is the member being specified.

@CURRMBR
@XREF

Index

Symbols

Numerics

A

B

C

D

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

E

F

G

H

I

L

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

M

N

O

P

Q

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

R

S

T

U

V

W

X

Z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Essbase Analytic Services

Release 7.1

Database Administrator's Guide

Volume II: Loading, Calculating, and Retrieving Data

Hyperion Solutions Corporation

Hyperion Solutions Corporation
1344 Crossman Avenue
Sunnyvale, California 94089

Printed in the U.S.A..

Contents

Part III

Building Dimensions and Loading Data

This part describes how to move data from external data sources into Analytic Services databases. It explains the concepts behind dimension builds and data loads; how to create rules files to manipulate the records, fields, and data in data sources; how to dynamically build or change dimensions in outlines; how to load data values into databases; and how to debug dimension builds and data loads. This part contains the following chapters:

- Chapter 16, "Understanding Data Loading and Dimension Building," explains data loading and dimension building, including the topics of external data sources, rules files, free-form data loading, and how Analytic Services handles security and multi-user issues.
- Chapter 17, "Creating Rules Files," describes how to create data load rules file and dimension build rules files, from opening the data source to validating and saving the rules file in Data Prep Editor.
- Chapter 18, "Using a Rules File to Perform Operations on Records, Fields, and Data," describes how to use a rules file to manipulate records, fields, and data values, including how to ignore fields, order fields, map fields to member names, and change data values.
- Chapter 19, "Performing and Debugging Data Loads or Dimension Builds," describes how to load data and build dimensions, from opening the data source and the rules file through performing the data load or dimension build to debugging problems that occur.
- Chapter 20, "Understanding Advanced Dimension Building Concepts," describes advanced topics, including detailed descriptions of build methods and discussions of building shared members, building attribute dimensions, and associating attributes.

Chapter 16

Understanding Data Loading and Dimension Building

An Analytic Services database contains dimensions, members, and data values.

- You can add data values, that is, numbers, to an Analytic Services database from a data source, such as a spreadsheet or a SQL database. This process is called loading data. If the data source is not perfectly formatted, you need a rules file to load the data values.
- You can add dimensions and members to an Analytic Services database manually, by using Outline Editor. You can also load dimensions and members into a database by using a data source and a rules file. This process is called building dimensions.

This chapter describes the components involved in loading data values and loading dimensions and members—data sources and rules files. This chapter contains the following sections:

Some rules file options and data source requirements vary for aggregate storage databases. See "Preparing Aggregate Storage Databases" on page 1325 for information about aggregate storage differences.

Process for Data Loading and Dimension Building

To load data values or dimensions and members into an Analytic Services database, follow these steps:

1. Set up the data source.

   If you are not using a rules file, you must set up the data source outside Analytic Services. For information on data sources, see "Data Sources" on page 356.

2. If necessary, set up the rules file.

   For a definition and discussion of rules files, see "Rules Files" on page 364.

3. Perform the data load or dimension build.

   For a comprehensive discussion of how to load data and members, see Chapter 19, "Performing and Debugging Data Loads or Dimension Builds."

Data Sources

Data sources contain the information that you want to load into the Analytic Services database. A data source can contain data values; information about members, such as member names, member aliases, formulas and consolidation properties; generation and level names; currency name and category; data storage properties; attributes; and UDAs (user-defined attributes).

The following sections describe the components of any kind of data source.

Supported Data Sources

Analytic Services supports the following types of data sources:

- Text files (flat files) from text backups or external sources.
- SQL data sources (non-Unicode only).
- Analytic Services export files. Export files do not need a rules file to load.
- Microsoft Excel files with the `.XLS` extension, Version 4.0 and higher.
- Microsoft Excel files, Version 5.0 and higher. Load as client objects or files in the file system.
- Spreadsheet audit log files.

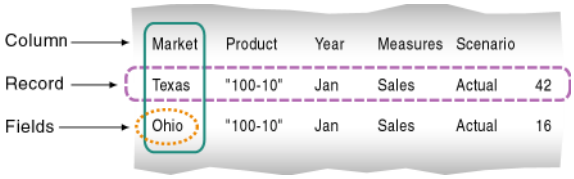
Note: If you are using Administration Services Console to load data or build an outline, spreadsheet files are not supported if the Administration Server is installed on a computer with a UNIX operating system.

Items in a Data Source

As illustrated in Figure 85, a data source is composed of records, fields, and field delimiters. A *record* is a structured row of related fields. A *field* is an individual value. A *delimiter* indicates that a field is complete and that the next item in the record is another field.

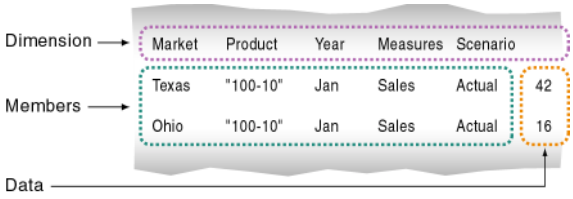
Analytic Services reads data sources starting at the top and proceeding from left to right.

Figure 85: Records and Fields



As illustrated in Figure 86, data sources can contain dimension fields, member fields, member combination fields, and data fields.

Figure 86: Kinds of Fields



- *Dimension fields* identify the dimensions of the database, such as Market. Use dimension fields to tell Analytic Services the order of the dimensions in the data source. In Figure 86, for example, the dimension fields are Market, Product, Year, Measures, and Scenario. Fields in the Market column, such as Texas, are members of the Market dimension, and fields in the Product column, such as 100-10, are members of the Product dimension. Although you can set dimension fields in the data source, usually you define dimension fields in the rules file.
- *Member fields* identify the members or member combinations of the specified dimensions. Use member fields to tell Analytic Services to which members to map new data values, or which members to add to the outline. In Figure 86, for example, Texas, 100-10, Jan, Sales, and Actual are all member fields.
- *Data fields* contain the numeric data values that are loaded into the intersections of the members of the database. Each data value must map to a dimension intersection. In Figure 86, for example, 42 is the data value that corresponds to the intersection of Texas, 100-10, Jan, Sales, and Actual.

  You can specify information in the header and in an individual record. In Figure 87, for example, 100 is the data value that corresponds to the intersection of Jan, Actual, Cola, East, Sales and 200 is the data value that corresponds to the intersection of Jan, Actual, Cola, West, Sales.

  Figure 87: Assigning Data Fields in Headers

```
Jan,  Actual
Cola   East   Sales   100
Cola   West   Sales   200
Cola   South  Sales   300
```

Data fields are used only for data loading; dimension builds ignore data fields. The following sections describe each item in a data source:

Valid Dimension Fields

In a data load, every dimension in the Analytic Services database must be specified in the either the data source or the rules file. If the data source does not identify every dimension in the database, you must identify the missing dimensions in a rules file. For example, the Sample Basic database has a dimension for Year. If several data sources arrive with monthly numbers from different regions, the month itself may not be specified in the data sources. You must specify the month in the data source header or the rules file. For information on setting header records, see "Defining Header Records" on page 392.

A dimension field must contain a valid dimension name. If you are not performing a dimension build, the dimension must already exist in the database. If you are performing a dimension build, the dimension name can be new, but the new name must be specified in the rules file.

Valid Member Fields

A member field can contain the name of a valid member or an alias. In Figure 86, for example, Texas and Ohio are valid members of the Market dimension. Analytic Services must know how to map each member field of the data source to a member of the database.

In order to be valid, a member field must meet the following criteria:

- The member field must contain a valid member name or member property. For a discussion of member properties, see "Using the Data Source to Set Member Properties" on page 379. If you are not performing a dimension build, the member must already exist in the outline. If you are performing a dimension build, the member can be new.

- Either the data source or the rules file must specify which dimension each member field maps to.
- A member field can map to a single member name, such as Jan (which is a member of the Year dimension), or to a member combination, such as Jan, Actual (which are members of the Year and Scenario dimensions).
- Member names that contain the same character as the file delimiter must be surrounded by double quotation marks. For example, if the data source is delimited by spaces, make sure that a member containing spaces, such as "New York," is surrounded by double quotation marks. If you are performing a data load without a rules file, member names containing some other characters must also be surrounded by quotation marks. For a list of the relevant characters, see "Data Sources That Do Not Need a Rules File" on page 365.

  Note: While processing each record in a data source for a data load, Analytic Services does not check to ensure a member specified in a member field belongs to the dimension specified for the dimension field. Analytic Services loads the data value to the data cell identified by the member combination in the record. In Figure 86, for example, if the second records reversed Jan and Sales (Texas, '100-10', Sales, Jan, Actual, 42), Analytic Services would load 42 to the correct data cell.

Valid Data Fields

If you are performing a dimension build, you can skip this section. Data fields are ignored during a dimension build.

Either the data source or the rules file must contain enough information for Analytic Services to determine where to put each data value. A data field contains the data value for its intersection in the database. In Figure 86, for example, 42 is a data field. It is the dollar sales of 100-10 (Cola) in Texas in January.

In a data field, Analytic Services accepts numbers and their modifiers, with no spaces or separators between them, and the text strings #MI and #MISSING.

Valid Modifiers	Examples
Currency symbols: • Dollar $ • Euro € • Yen ¥	$12 is a valid value. $ 12 is not a valid value because there is a space between the dollar sign and the 12.
Parentheses around numbers to indicate a negative number	(12)
Minus sign before numbers. Minus signs after numbers are *not* valid.	-12
Decimal point	12.3
Large numbers with or without commas	Both 1,345,218 and 1345218 are valid values.
#MI or #MISSING to represent missing or unknown values	You must insert #MI or #MISSING into any data field that has no value. If you do not, the data source may not load correctly. For instructions on how to replace a blank field with #MI or #MISSING, see "Replacing an Empty Field with Text" on page 400.

If the data source contains a member field for every dimension and one field that contains data values, you must define the field that contains data values as a data field in the rules file. To read Figure 88 into the Sample Basic database, for example, define the last field as a data field.

Figure 88: Setting Data Fields

```
Jan     Cola     East     Sales     Actual     100
Feb     Cola     East     Sales     Actual     200
```

➤ To define a data field, see "Defining a Column as a Data Field" in *Essbase Administration Services Online Help*.

If there is no value in the data field (or the value is #MISSING), Analytic Services does not change the existing data value in the database. Analytic Services does not replace current values with empty values.

Note: If the data source contains blank fields for data values, replace them with #MI or #MISSING. Otherwise, the data may not load correctly. For instructions on how to replace a blank field with #MI or #MISSING, see "Replacing an Empty Field with Text" on page 400.

Valid Delimiters

You must separate fields from each other with delimiters. If you are loading data without a rules file, you must uses spaces to delimit fields.

If you are using a rules file, delimiters can be any of the following:

- Tabs (Tabs are the default delimiter expected by Analytic Services.)
- Spaces
- New lines
- Carriage returns
- Commas

Extra Delimiters Without a Rules File

In data sources that are loaded without a rules file, Analytic Services ignores extra delimiters. In Figure 89, for example, the fields are separated by spaces. Analytic Services ignores the extra spaces between the fields.

Figure 89: File Delimiters

```
East      Cola   Actual   Jan   Sales   10
East   Cola   Actual   Feb   Sales   21
East   Cola   Actual   Mar   Sales   30
```

Extra Delimiters with a Rules File

In data sources that are loaded with a rules file, Analytic Services reads extra delimiters as empty fields. For example, if you try to use a rules file to load the file in Figure 90 into the Sample Basic database, the load fails. Analytic Services reads the extra comma between East and Cola in the first record as an extra field. Analytic Services then puts Cola into Field 3. In the next record, however, Cola is in Field 2. Analytic Services expects Cola to be in Field 3 and stops the data load.

Figure 90: File Delimiters

```
East,,Cola,Actual,Jan,Sales,10
East,Cola,Actual,Feb,Sales,21
East,Cola,Actual,Mar,Sales,30
```

To solve the problem, delete the extra delimiter from the data source.

Valid Formatting Characters

Analytic Services views some characters in the data source as formatting characters only. For that reason, Analytic Services ignores the following characters:

==	Two or more equal signs, such as for double underlining
--	Two or more minus signs, such as for single underlining
__	Two or more underscores
==	Two or more IBM PC graphic double underlines (ASCII character 205)
__	Two or more IBM PC graphic single underlines (ASCII character 196)

Ignored fields do not affect the data load or dimension build.

For example, Analytic Services ignores the equal signs in Figure 91 and loads the other fields normally.

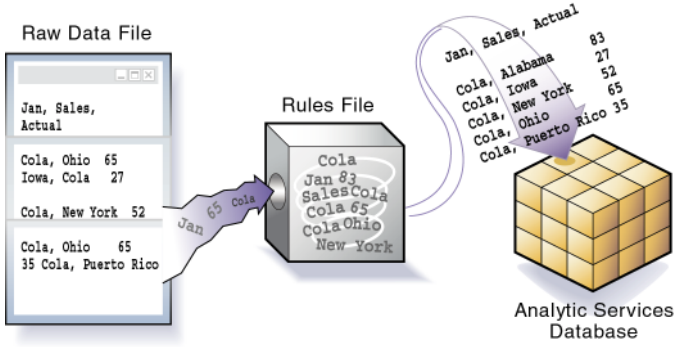
Figure 91: Ignoring Formatting Characters During Loading

```
East      Actual    "100-10"
          Sales     Marketing
          =====     =========
Jan        10           8
Feb        21          16
```

Rules Files

Rules are a set of operations that Analytic Services performs on data values or on dimensions and members when it processes a data source. Use rules to map data values to an Analytic Services database or to map dimensions and members to an Analytic Services outline.

Figure 92: Loading Data Sources Through Rules Files



Rules are stored in rules files. A rules file tells Analytic Services which build method to use, specifies whether data values or members are sorted or in random order, and tells Analytic Services how to transform data values or members before loading them. It is best to create a separate rules file for each dimension.

Analytic Services reads the data values or members in the data source, changes them based on the rules in the rules file, and loads the changed data values into the database and the changed members into the outline. Analytic Services does not change the data source. You can re-use a rules file with any data source that requires the same set of rules.

After you create a dimension build rules file, you may want to automate the process of updating dimensions. See Appendix D, “Using ESSCMD.”

Situations That Do and Do Not Need a Rules File

You need a rules file if the data source does not map perfectly to the database or if you are performing any of the following tasks:

- Loading data from a SQL data source
- Building dimensions
  - Adding new dimensions and members to the database
  - Changing existing dimensions and members in the database
- Changing the data in any way, including the following:
  - Ignoring fields or strings in the data source
  - Changing the order of fields by moving, joining, splitting, or creating fields
  - Mapping the data in the data source to the database by changing strings
  - Changing the data values in the data source by scaling data values or by adding data values to existing data values in the data source
  - Setting header records for missing values
  - Rejecting an invalid record and continuing the data load

You do not need a rules file if you are performing a data load and the data source maps perfectly to the database. For a description of a data source that maps perfectly, see "Data Sources That Do Not Need a Rules File" on page 365.

Note: If you are using a rules file, each record in the rules file must have the same number of fields. See "Dealing with Missing Fields in a Data Source" on page 409.

Data Sources That Do Not Need a Rules File

If you are performing a dimension build, skip this section. You cannot perform a dimension build without a rules file.

If a data source contains all information required to load the data values in it into the database, you can load the data source directly. This kind of load is called a *free-form data load*.

To load a data value successfully, Analytic Services must encounter one member from each dimension before encountering the data value. For example, in Figure 86, Analytic Services loads the data value 42 into the database with the members Texas, 100-10, Jan, Sales, and Actual. If Analytic Services encounters a data value before a member of each dimension is specified, it stops loading the data source.

To map perfectly, a data source must contain all of the following and nothing other than the following:

- One or more valid members from each dimension. A member name must be enclosed in quotation marks if it contains any of the following:
  - Spaces
  - Numeric characters (0–9)
  - Dashes (minus signs, hyphens)
  - Plus signs
  - & (ampersands)

  If you are performing a data load without a rules file, when Analytic Services encounters an invalid member field, it stops the data load. Analytic Services loads all fields read before the invalid field into the database, resulting in a partial load of the data values. For information on continuing the load, see "Loading Dimension Build and Data Load Error Logs" on page 1018.
- One or more valid data values. For examples of valid values, see "Valid Data Fields" on page 360.

  If the data source contains blank fields for data values, replace the blank fields with #MI or #MISSING. Otherwise, the data values may not load correctly.
- Valid delimiters. For a list of valid delimiters, see "Valid Delimiters" on page 362.

The fields in the data source must be formatted in an order that Analytic Services understands. The simplest way to format a record is to include a member from each dimension and a data field, as illustrated in Figure 93:

Figure 93: Sample Free-Form Data Source

```
Sales "100-10" Ohio Jan Actual 25
Sales "100-20" Ohio Jan Actual 25
Sales "100-30" Ohio Jan Actual 25
```

If the data source is not correctly formatted, it will not load. You can edit the data source using a text editor and fix the problem. If you find that you must perform many edits (such as moving several fields and records), it might be easier to use a rules file to load the data source. For a definition and discussion of rules files, see "Rules Files" on page 364.

The following sections describe more complicated ways to format free-form data sources:

- "Formatting Ranges of Member Fields" on page 367
- "Formatting Columns" on page 370

Formatting Ranges of Member Fields

If you are performing a dimension build, skip this section. You cannot perform a dimension build without a rules file.

You can express member names as ranges within a dimension. For example, Sales and COGS form a range in the Measures dimension. Ranges of member names can handle a series of values.

A data source can contain ranges from more than one dimension at a time.

In Figure 94, for example, Jan and Feb form a range in the Year dimension and Sales and COGS form a range in the Measures dimension.

Figure 94: Multiple Ranges of Member Names

```
Actual Texas      Sales        COGS
                 Jan    Feb    Jan   Feb
"100-10"         98     89     26    19
"100-20"         87     78     23    32
```

In Figure 94, Sales is defined for the first two columns and COGS for the last two columns.

The following sections describe additional types of ranges:

- "Setting Ranges Automatically" on page 368
- "Handling Out of Range Data Values" on page 368
- "Interpreting Duplicate Members in a Range" on page 369
- "Reading Multiple Ranges" on page 370

Setting Ranges Automatically

If you are performing a dimension build, skip this section. You cannot perform a dimension build without a rules file.

When Analytic Services encounters two or more members from the same dimension with no intervening data fields, it sets up a range for that dimension. The range stays in effect until Analytic Services encounters another member name from the same dimension, at which point Analytic Services replaces the range with the new member or new member range.

Figure 95, for example, contains a range of Jan to Feb in the Year dimension. It remains in effect until Analytic Services encounters another member name, such as Mar. When Analytic Services encounters Mar, the range changes to Jan, Feb, Mar.

Figure 95: Ranges of Member Names

```
Texas Sales
                      Jan    Feb    Mar
Actual     "100-10"   98     89     58
           "100-20"   87     78     115
```

Handling Out of Range Data Values

If you are performing a dimension build, skip this section. You cannot perform a dimension build without a rules file.

When Analytic Services encounters a member range, it assumes that there is a corresponding range of data values. If the data values are not in the member range, the data load stops. Analytic Services loads any data fields read before the invalid field into the database, resulting in a partial load of the data.

Figure 96, for example, contains more data fields than member fields in the defined range of members. The data load stops when it reaches the 10 data field. Analytic Services loads the 100 and 120 data fields into the database.

Figure 96: Extra Data Values

```
Cola     Actual    East
           Jan     Feb
Sales      100     120     10
COGS       30      34      32
```

For information on restarting the load, see “Loading Dimension Build and Data Load Error Logs” on page 1018.

Interpreting Duplicate Members in a Range

If you are performing a dimension build, skip this section. You cannot perform a dimension build without a rules file.

Be sure to structure ranges in the source data so that Analytic Services interprets them correctly. If the a member appears more than once in a range, Analytic Services ignores the duplicates.

The file in Figure 97 contains two ranges: Actual to Budget and Sales to COGS. It also contains duplicate members.

Figure 97: Duplicate Members in a Range

```
Cola East
        Actual    Budget    Actual    Budget
        Sales     Sales     COGS      COGS
Jan     108       110       49        50
Feb     102       120       57        60
```

Analytic Services ignores the duplicate members. The members that Analytic Services ignores have a line through them in the following example:

Figure 98: Ignored Duplicate Members

```
Cola East
        Actual    Budget    ~~Actual~~    ~~Budget~~
        Sales     ~~Sales~~     COGS      ~~COGS~~
Jan     108       110       49        50
Feb     102       120       57        60
```

For Actual, the first member of the first range, Analytic Services maps data values to each member of the second range (Sales and COGS). Analytic Services then proceeds to the next value of the first range, Budget, similarly mapping values to each member of the second range. As a result, Analytic Services interprets the file as shown in Figure 99.

Figure 99: How Analytic Services Interprets the File in Figure 97

```
Cola East
           Actual              Budget
           Sales     COGS      Sales     COGS
Jan        108       110       49        50
Feb        102       120       57        60
```

Reading Multiple Ranges

If you are performing a dimension build, skip this section. You cannot perform a dimension build without a rules file.

As Analytic Services scans a file, it processes the most recently encountered range first when identifying a range of data values. In Figure 99, for example, there are two ranges: Actual and Budget and Sales and COGS. While reading the file from left to right and top to bottom, Analytic Services encounters the Actual and Budget range first and the Sales and COGS range second. Because the Sales and COGS range is encountered second, Analytic Services puts data fields in the Sales and COGS part of the database first.

Formatting Columns

If you are performing a dimension build, skip this section. You cannot perform a dimension build without a rules file.

Files can contain columns of fields. Columns can be symmetric or asymmetric. Symmetric columns have the same number of members under them. Asymmetric columns have different numbers of members under them. Analytic Services supports loading data from both types of columns.

Symmetric Columns

If you are performing a dimension build, skip this section. You cannot perform a dimension build without a rules file.

Symmetric columns have the same number of members under them. In Figure 100, for example, each dimension column has one column of members under it. For example, Product has one column under it (100-10 and 100-10) and Market has one column under it (Texas and Ohio).

Figure 100: Symmetric Columns

```
Product    Measures    Market    Year    Scenario
"100-10"   Sales       Texas     Jan     Actual      112
"100-10"   Sales       Ohio      Jan     Actual      145
```

The columns in the following file are also symmetric, because Jan and Feb have the same number of members under them:

Figure 101: Groups of Symmetric Columns

```
                                Jan           Feb
                           Actual  Budget  Actual  Budget
"100-10"  Sales  Texas     112     110     243     215
"100-10"  Sales  Ohio      145     120     81      102
```

Asymmetric Columns

If you are performing a dimension build, skip this section. You cannot perform a dimension build without a rules file.

Columns can also be asymmetric. In Figure 102, the Jan and Feb columns are asymmetric because Jan has two columns under it (Actual and Budget) and Feb has only one column under it (Budget):

Figure 102: Valid Groups of Asymmetric Columns

```
                           Jan     Jan     Feb
                           Actual  Budget  Budget
"100-10"  Sales  Texas     112     110      243
"100-10"  Sales  Ohio      145     120      81
```

If a file contains asymmetric columns, you must label each column with the appropriate member name.

The file in Figure 103, for example, is not valid because the column labels are incomplete. The Jan label must appear over both the Actual and Budget columns.

Figure 103: Invalid Asymmetric Columns

```
                           Jan             Feb
                           Actual  Budget  Budget
"100-10"  Sales  Texas     112     110     243
"100-10"  Sales  Ohio      145     120     81
```

This file in Figure 104 is valid because the Jan label is now over both Actual and Budget. It is clear to Analytic Services that both of those columns map to Jan.

Figure 104: Valid Asymmetric Columns

```
                           Jan     Jan     Feb
                           Actual  Budget  Budget
"100-10"  Sales  Texas     112     110      243
"100-10"  Sales  Ohio      145     120      81
```

Security and Multiple-User Considerations

Analytic Services supports concurrent multiple users reading and updating the database. Thus, users can use the database while you are dynamically building dimensions, loading data, or calculating the database. In a multi-user environment, Analytic Services protects data by using the security system described in Chapter 36, "Managing Security for Users and Applications".

- Security Issues

  The security system prevents unauthorized users from changing the database. Only users with write access to a database can load data values or add dimensions and members to the database. Write access can be provided globally or by using filters.

- Multi-User Data Load Issues

  You can load data values while multiple users are connected to a database. Analytic Services uses a block locking scheme for handling multi-user issues. When you load data values, Analytic Services does the following:

  – Locks the block it is loading into so that no one can write to the block.

    See Chapter 46, "Ensuring Data Integrity" for information on Analytic Services transaction settings, such as identifying whether other users get read-only access to the locked block or noting how long Analytic Services waits for a locked block to be released.

  – Updates the block.

    See "Data Locks" on page 1054 for information on whether Analytic Services unlocks a block when its update is complete or waits for the entire data load to complete before unlocking the block.

- Multi-User Dimension Build Issues

  You cannot build dimensions while other users are reading or writing to the database. After you build dimensions, Analytic Services restructures the outline and locks the database for the duration of the restructure operation.

Chapter 17

Creating Rules Files

A rules files tells Analytic Services what changes to make to the data source and outline during a data load or dimension build. For a definition and discussion of rules files, see "Rules Files" on page 364. This chapter describes how to create a rules file for data loading or dimension building:

For a comprehensive discussion of performing a data load or dimension build after you create a rules file, see Chapter 19, "Performing and Debugging Data Loads or Dimension Builds."

Understanding the Process for Creating Data Load Rules Files

To create a data load rules file, follow these steps:

1. Determine whether to use the same rules file for data loading and dimension building.

   For a discussion of factors that influence your decision, see "Combining Data Load and Dimension Build Rules Files" on page 376.

2. Create a new rules file.

   For a process map, see "Creating Rules Files" on page 376.

3. Set the file delimiters for the data source.

   For a description of file delimiters, see "Setting File Delimiters" on page 377.

4. If necessary, set record, field, and data operations to change the data in the data source during loading.

   For a comprehensive discussion, see Chapter 18, "Using a Rules File to Perform Operations on Records, Fields, and Data."

5. Validate and save the rules file.

   For references for pertinent topics, see "Validating, Saving, and Printing" on page 386.

For a comprehensive discussion of data sources and rules files, see Chapter 16, "Understanding Data Loading and Dimension Building."

Understanding the Process for Creating Dimension Build Rules Files

To create a dimension build rules file, follow these steps:

1. Determine whether to use the same rules file for data loading and dimension building.

   For a discussion of factors that influence your decision, see "Combining Data Load and Dimension Build Rules Files" on page 376.

2. Create a new rules file.

   For a process map, see "Creating Rules Files" on page 376.

3. Set the file delimiters for the data source.

   For a description of file delimiters, see "Setting File Delimiters" on page 377.

4. If you are creating a new dimension, name the dimension.

   For references to pertinent topics, see "Naming New Dimensions" on page 378.

5. Select the build method.

   For references to pertinent topics, see "Selecting a Build Method" on page 378.

6. If necessary, change or set the properties of members and dimensions you are building.

   For references to pertinent topics, see "Setting and Changing Member and Dimension Properties" on page 379.

7. If necessary, set record and field operations to change the members in the data source during loading.

   For a comprehensive discussion, see Chapter 18, "Using a Rules File to Perform Operations on Records, Fields, and Data."

8. Set field type information, including field type, field number, and dimension.

   For references to pertinent topics, see "Setting Field Type Information" on page 381.

9. Validate and save the rules file.

   For references to pertinent topics, see "Validating, Saving, and Printing" on page 386.

For a comprehensive discussion of data sources and rules files, see Chapter 16, "Understanding Data Loading and Dimension Building."

Combining Data Load and Dimension Build Rules Files

Before you start building a rules file, you should determine whether to use that rules file for both data load and dimension build. Once you create a rules file, you cannot separate it into two rules files. Likewise, once you create two rules files, you cannot merge them into one rules file.

Use the same rules file for both data load and dimension build if you wish to load the data source and build new dimensions at the same time.

Use separate rules files for data load and dimension build under any of the following circumstances:

- To build an outline from scratch
- To perform different field operations during the data load and dimension build
- To re-use the data load or dimension build rules file separately
- To use data sources that contain no data values, only dimensions

Creating Rules Files

To create a new rules file:

1. If you are creating the rules file on the Analytic Server, connect to the server. If you are creating the rules file on the client, you do not need to connect to the Analytic Server.
2. Open Data Prep Editor.

   For references to pertinent topics, see “Opening Data Prep Editor” on page 377.
3. Open the data source.

   For a brief discussion and for references to pertinent topics, see “Opening a Data Source” on page 377.

Opening Data Prep Editor

You can open Data Prep Editor with a new or existing rules file. After you open Data Prep Editor, be sure to put the editor in the correct mode.

- To open Data Prep Editor, see "Creating a Rules File" or "Opening an Existing Rules File" in the *Essbase Administration Services Online Help*.
- To learn how to use Data Prep Editor, see "About Data Prep Editor" in the *Essbase Administration Services Online Help*.

Opening a Data Source

After you open Data Prep Editor, you can open data sources, such as text files, spreadsheet files, and SQL data sources. The data source appears in Data Prep Editor so that you can see what needs to be changed.

You can open a SQL data source only if you have licensed Essbase SQL Interface. The *Essbase Analytic Services SQL Interface Guide* provides information on supported environments, installation, and connection to supported data sources. Contact your Analytic Services administrator for more information. When you open a SQL data source, the rules fields default to the column names of the SQL data source. If the names are not the same as the Analytic Services dimension names, you need to map the fields to the dimensions. For a comprehensive discussion of mapping, see "Changing Field Names" on page 399.

- To open text files and spreadsheet files, see "Opening a Data File" in the *Essbase Administration Services Online Help*.
- To open SQL data sources, see "Opening a SQL Data Source" in the *Essbase Administration Services Online Help*.

Setting File Delimiters

A file delimiter is the character (or characters) used to separate fields in the data source. By default, a rules file expects fields to be separated by tabs. You can set the file delimiter expected to be a comma, tab, space, fixed-width column, or

custom value. Acceptable custom values are characters in the standard ASCII character set, numbered from 0 through 127. Usually, setting the file delimiters is the first thing you do after opening a data source.

Note: You do not need to set file delimiters for SQL data.

➤ To set file delimiters, see "Setting File Delimiters" in the *Essbase Administration Services Online Help*.

Naming New Dimensions

If you are not creating a new dimension in the rules file, skip this section.

If you are creating a new dimension, you must name it in the rules file. Before choosing a dimension name, see "Understanding the Rules for Naming Dimensions and Members" on page 143.

If you are creating an attribute dimension, the base dimension must be a sparse dimension already defined in either the outline or the rules file. For a comprehensive discussion of attribute dimensions, see Chapter 10, "Working with Attributes."

➤ To name a new dimension, see "Creating a New Dimension Using a Rules File" in the *Essbase Administration Services Online Help*.

Selecting a Build Method

If you are not performing a dimension build, skip this section.

If you are building a new dimension or adding members to an existing dimension, you must tell Analytic Services what algorithm, or build method, to use. You must specify a build method for each dimension that you are creating or modifying. For information about each build method, see Table 22 on page 420.

➤ To select a build method, see "Choosing a Build Method" in the *Essbase Administration Services Online Help*.

Setting and Changing Member and Dimension Properties

If you are not performing a dimension build, skip this section.

If you are performing a dimension build, you can set or change the properties of the members and dimensions in the outline. Some changes affect all members of the selected dimension, some affect only the selected dimension, and some affect all dimensions in the rules file.

You can set or change member and dimension properties using the Data Prep Editor or a change the member properties in the data source.

Using the Data Prep Editor to Set Dimension and Member Properties

- To set dimension properties, see "Setting Dimension Properties" in the *Essbase Administration Services Online Help*.
- To set member properties, see "Setting Member Properties" in the *Essbase Administration Services Online Help*.

Using the Data Source to Set Member Properties

You can modify the properties of both new and existing members during a dimension build by including member properties in a field in the data source. In the data source, put the properties in the field directly following the field containing the members that the properties modify. For example, to specify that the Margin% member not roll up into its parent and not be shared.

1. Position the ~ property (which indicates that the member should not roll up into its parent) and the N property (which indicates that the member should not be shared) after the Margin% field:

   ```
   Margin%  Margin%  ~ N Sales
   ```

2. Set the field type for the properties fields to Property. For a brief discussion and pertinent references, see "Setting Field Type Information" on page 381.

The following table lists all member codes used in the data source to assign properties to block storage outline members. For a list of properties that can be assigned to aggregate storage outline members, see "Rules File Differences for Aggregate Storage Dimension Builds" on page 1326.

Code	Description
%	Express as a percentage of the current total in a consolidation
*	Multiply by the current total in a consolidation
+	Add to the current total in a consolidation
-	Subtract from the current total in a consolidation
/	Divide by the current total in a consolidation
~	Exclude from the consolidation
A	Treat as an average time balance item (applies to accounts dimensions only)
B	Exclude data values of zero or #MISSING in the time balance (applies to accounts dimensions only)
E	Treat as an expense item (applies to accounts dimensions only)
F	Treat as a first time balance item (applies to accounts dimensions only)
L	Treat as a last time balance item (applies to accounts dimensions only)
M	Exclude data values of #MISSING from the time balance (applies to accounts dimensions only)
N	Never allow data sharing
O	Tag as label only (store no data)
S	Set member as stored member (non-Dynamic Calc and not label only)
T	Require a two-pass calculation (applies to accounts dimensions only)
V	Create as Dynamic Calc and Store
X	Create as Dynamic Calc
Z	Exclude data values of zero from the time balance (applies to accounts dimensions only)

Performing Operations on Records, Fields, and Data

In a rules file you can perform operations on records, fields, and data values before loading them into the database. The data source is not changed.

For a comprehensive discussion, see Chapter 18, "Using a Rules File to Perform Operations on Records, Fields, and Data."

Setting Field Type Information

If you are not performing a dimension build, skip this section.

In a dimension build, each field in the data source is part of a column that describes a member in the outline. Fields can contain information about member names, member properties, or attribute associations. In order for Analytic Services to process this information, you must specify the field type in the rules file. You must specify the following information when setting field types:

- The type of field to expect in that column, such as a generation field or an alias field. The field type to choose depends on the data source and the build method. See "Understanding Build Methods" on page 419.
- The dimension to which the members of that column belong. See "List of Field Types" on page 382, "Rules for Field Types" on page 384 and "Using the Data Source to Set Member Properties" on page 379.
- The generation or level number of the members of that column.

The following sections contain detailed information about field types:

- "List of Field Types" on page 382
- "Rules for Field Types" on page 384
- "Using the Data Source to Set Member Properties" on page 379

➤ To set field information, see "Setting Field Types" in the *Essbase Administration Services Online Help.*

List of Field Types

Table 20 lists valid field types for each build method.

Table 20: Field Types

Field Type	What the Field Contains	Valid Build Methods
Alias	An alias **Note:** If the Member update dimension build setting is set to Remove unspecified and the data source for a new member contains the alias value of a removed member, the alias value will not be assigned to the new member.	Generation, level, and parent-child references
Property	A member property. For a list of properties to set in the data source, see "Using the Data Source to Set Member Properties" on page 379.	
Formula	A formula	
Currency name	A currency name (block storage outlines only)	
Currency category	A currency category ((block storage outlines only)	
UDA	A UDA (user-defined attribute)	
Attribute parent	In an attribute dimension, the name of the parent member of the attribute member in the following field	
The name of a specific attribute dimension	A member of the specified attribute dimension. This member will be associated with a specified generation or level of the selected base dimension.	

Table 20: Field Types (Continued)

Field Type	What the Field Contains	Valid Build Methods
Generation	The name of a member in the specified generation	Generation references
Duplicate generation	The name of a member that has duplicate parents; that is, a member that is shared by more than one parent	
Duplicate generation alias	The alias for the shared member	
Level	The name of a member in a level	Level references
Duplicate level	The name of a member that has duplicate parents; that is, a member that is shared by more than one parent	
Duplicate level alias	The alias for the shared member	
Parent	The name of a parent	Parent-child reference
Child	The name of a child	

Rules for Field Types

The field type that you choose for a field depends on the build method that you selected. Table 21 lists the rules for selecting valid field types, depending on the build method. If necessary, move the fields to the required locations. For a brief discussion, see "Moving Fields" on page 396.

➤ To move fields, see "Moving Fields" in the *Essbase Administration Services Online Help*.

Table 21: Field Numbers

Build Method	Rules for Assigning Field Types

Generation	• If GEN numbers do not start at 2, the first member of the specified generation must exist in the outline.
	• GEN numbers must form a contiguous range. For example, if GEN 3 and GEN 5 exist, you must also define GEN 4.
	• Put DUPGEN fields immediately after GEN fields.
	• Put DUPGENALIAS fields immediately after DUPGEN fields.
	• Group GEN fields sequentially within a dimension; for example:
	`GEN2,PRODUCT GEN3,PRODUCT GEN4,PRODUCT`
	• Put attribute association fields after the base field with which they are associated and specify the generation number of the associated base dimension member; for example:
	`GEN2,PRODUCT GEN3,PRODUCT OUNCES3,PRODUCT`
	The generation number must correspond to the generation of the member in the outline for which the field provides values. For example, the 3 in GEN3,PRODUCT shows that the values in the field are third generation members of the Product dimension. The 2 in ALIAS2,POPULATION shows that the values in the field are associated with the second generation member of the Population dimension.

Table 21: Field Numbers

Build Method	Rules for Assigning Field Types
Level	• Put DUPLEVEL fields immediately after LEVEL fields. • Put DUPLEVELALIAS fields immediately after the DUPLEVEL fields. • Each record must contain a level 0 member. If a level 0 member is repeated on a new record with a different parent, Analytic Services rejects the record unless you select the Allow Moves member property. To set member properties, see "Setting Member Properties" in the *Essbase Administration Services Online Help*. • Group level fields sequentially within a dimension. • Put the fields for each roll-up in sequential order. • Use a single record to describe the primary and secondary roll-ups. • Put attribute association fields after the base field with which they are associated and specify the level number of the associated base dimension member; for example: `LEVEL3,PRODUCT  UNCES3,PRODUCT  LEVEL2,PRODUCT` • The level number must correspond to the level of the member in the outline for which the field provides values. For example, the 3 in LEVEL3,PRODUCT shows that the values in the field are level 3 members of the Product dimension. The 2 in ALIAS2,POPULATION shows that the values in the field are associated with the second level of the Population dimension.
Parent-child	If field type is parent or child, enter 0 (zero) in the Number text box.
Attribute dimension name	The generation or level number must correspond to the generation or level of the associated base member in the outline. For example, the 3 in OUNCES3,PRODUCT shows that the values in the field are the members of the Ounces attribute dimension that are associated with the third generation member of the Product dimension in the same source data record.

Validating, Saving, and Printing

Rules files are validated to make sure that the members and dimensions in the rules file map to the outline. Validation cannot ensure that the data source loads properly.

➤ To validate a rules file, see "Validating a Rules File" in the *Essbase Administration Services Online Help*.

➤ To save a rules file, see "Saving a Rules File" in the *Essbase Administration Services Online Help*.

If the rules file is not valid, complete one of the following actions:

- If you are validating a data load rules file, see "Requirements for Valid Data Load Rules Files" on page 386.
- If you are validating a dimension build rules file, see "Requirements for Valid Dimension Build Rules Files" on page 387.

If the rules file is correct, you can perform a data load or dimension build. For a comprehensive discussion of how to load data and members, see Chapter 19, "Performing and Debugging Data Loads or Dimension Builds."

Requirements for Valid Data Load Rules Files

For a data load rules file to validate, all the following questions must be answered "yes."

- Is the rules file associated with the correct outline? For information on associating rules files with outlines, see "Validating a Rules File" in the *Essbase Administration Services Online Help*.
- Does each record in the data source contain only one member from each dimension? For a brief discussion, see "Items in a Data Source" on page 357.
- Are all member and dimension names spelled correctly?
- Are all members surrounded by quotation marks if they contain numbers or file delimiters? For a description of member names that require quotation marks, see "Valid Member Fields" on page 359.
- Are there no extra delimiters in the data source? For a discussion of the problems created by extra delimiters, see "Extra Delimiters with a Rules File" on page 363.

- Is the member that to which each data field maps spelled correctly in the rules file? For information on mapping data fields, see "Changing Field Names" on page 399.
- Are the file delimiters correctly placed? For a discussion of the placement of delimiters, see "Valid Delimiters" on page 362.
- Is the member in the field name a valid member? For information on setting field names, see "Mapping Fields" on page 399.
- Is the dimension name used in only a single field, that is, not in a field name and the header? You can map a single data value to only one set of members.
- Is only one field defined as a data field? For a brief discussion and a reference to instructions, see "Defining a Column as a Data Field" on page 402.
- Is the UDA used for sign flipping in the associated outline? For a brief discussion, see "Flipping Field Signs" on page 404.

Requirements for Valid Dimension Build Rules Files

For a dimension build rules file to validate, all of the following questions must be answered "yes."

- Is the rules file associated with the correct outline? For information on associating rules files with outlines, see "Validating a Rules File" in the *Essbase Administration Services Online Help*.
- Does each record contain only one member from each dimension? For a brief discussion, see "Items in a Data Source" on page 357.
- Are all member and dimension names spelled correctly?
- Are all members surrounded by quotation marks if they contain numbers or file delimiters? For a description of member names that require quotation marks, see "Valid Member Fields" on page 359.
- Are there no extra delimiters in the data source? For a discussion of the problems created by extra delimiters, see "Extra Delimiters with a Rules File" on page 363.
- Are the reference numbers sequential? For a discussion of limitations and obligations, see "Rules for Field Types" on page 384.
- Are there no repeated generations? For a discussion of limitations and obligations, see "Rules for Field Types" on page 384.

- Is each field type valid for the build method? See "List of Field Types" on page 382 to identify which field type can be used with which build methods.
- Are all the fields in correct order? For a review of the rules for selecting valid field types, see "Rules for Field Types" on page 384.
- Does each child field have a parent field?
- Do all dimension names exist in the outline or the rules file?
- Are any dimensions specified in both the header record in the rules file and the header record in the data source? Dimensions can be specified in either the header in the rules file or the header in the data source, but not in both. For a discussion of header records, see "Defining Header Records" on page 392.

Copying Rules Files

You can copy rules files to applications and databases on any Analytic Server, according to your permissions. You can also copy rules files across servers as part of application migration.

➤ To copy a rules file, use any of the following methods:

Tool	Topic	Location
Administration Services	Copying a Rules File	*Essbase Administration Services Online Help*
MaxL	**alter object**	*Technical Reference*
ESSCMD	COPYOBJECT	*Technical Reference*

Printing Rules Files

You can print the entire contents and properties of a data load or dimension build rules file. You can also specify properties and settings to print.

➤ To print a rules file, see "Printing Rules Files" in *Essbase Administration Services Online Help*.

Chapter 18

Using a Rules File to Perform Operations on Records, Fields, and Data

This chapter describes how to edit a rules file to perform operations on records, fields, and data before loading the database. For a comprehensive discussion of data sources and rules files, see Chapter 16, "Understanding Data Loading and Dimension Building." For a comprehensive discussion of the process of creating a rules file, see Chapter 17, "Creating Rules Files."

This chapter contains the following sections about record operations:

This chapter contains the following sections about field operations:

This chapter contains the following sections about data operations:

Performing Operations on Records

You can perform operations at the record level. For example, you can reject certain records before they are loaded into the database.

This section contains the following sections:

Selecting Records

You can specify which records Analytic Services loads into the database or uses to build dimensions by setting selection criteria. *Selection criteria* are string and number conditions that must be met by one or more fields within a record before Analytic Services loads the record. If a field or fields in the record do not meet the selection criteria, Analytic Services does not load the record. You can define one or more selection criteria. For example, to load only 2003 Budget data from a data source, create a selection criterion to load only records in which the first field is Budget and the second field is 2003.

➤ To select a record, see "Selecting Records" in the *Essbase Administration Services Online Help*.

Note: If you define selection criteria on more than one field, you can specify how Analytic Services combines the criteria. For a brief discussion, see "Combining Multiple Select and Reject Criteria" on page 391.

Rejecting Records

You can specify which records Analytic Services ignores by setting rejection criteria. *Rejection criteria* are string and number conditions that, when met by one or more fields within a record, cause Analytic Services to reject the record. You can define one or more rejection criteria. If no field in the record meets the rejection criteria, Analytic Services loads the record. For example, to reject Actual data from a data source and load only Budget data, create a rejection criterion to reject records in which the first field is Actual.

➤ To reject a record, see “Rejecting Records” in the *Essbase Administration Services Online Help*.

Note: If you define rejection criteria on more than one field, you can specify how Analytic Services should combine the criteria. For a brief discussion, see “Combining Multiple Select and Reject Criteria” on page 391.

Combining Multiple Select and Reject Criteria

When you define select and reject criteria on multiple fields, you can specify how Analytic Services combines the rules across fields, that is, whether the criteria are connected logically with AND or with OR. If you select And from the Boolean group, the fields must match all of the criteria. If you select Or from the Boolean group, the fields must match only one of the criteria. The global Boolean setting applies to all select or reject operations in the rules file, for both data load and dimension build fields.

Note: If selection and rejection criteria apply to the same record (that is, you try to select and reject the same record), the record is rejected.

➤ To determine how to combine select and reject criteria on multiple fields, see “Combining Selection and Rejection Criteria” in the *Essbase Administration Services Online Help*.

Setting the Records Displayed

You can specify the number of records that Analytic Services displays in Data Prep Editor. You can also specify the first record in Data Prep Editor. Analytic Services skips all preceding records and, in Data Preparation Editor, begin the display with the record number you chose as first. For example, if you enter `5` as the starting record, Analytic Services does not display records 1 through 4.

Note: Analytic Services treats header records the same as data records when counting the records to skip.

➤ To set the records displayed, see “Setting the Records Displayed” in the *Essbase Administration Services Online Help*.

Defining Header Records

Data sources can contain data records and header records. *Data records* contain member fields and data fields. *Header records* describe the contents of the data source and describe how to load values from the data source to the database.

Rules files contain records that translate the data of the data source to map it to the database. As part of that information, rules files can also contain header records. For example, the Sample Basic database has a dimension for Year. If several data sources arrive with monthly numbers from different regions, the month itself might not be specified in the data sources. You must set header information to specify the month.

You can create a header record using either of the following methods:

- You can define header information in the rules file. Rules file headers are used only during data loading or dimension building and do not change the data source. Header information set in a rules file is not used if the rules file also points to header records in the data source.
- You can define header information in the data source by using a text editor or spreadsheet and then pointing to the header records in the rules file. Placing header information in the data source makes it possible to use the same rules file for multiple data sources with different formats, because the data source format is specified in the data source header and not in the rules file.

  When you add one or more headers to the data source, you must also specify the location of the headers in the data source in the rules file. The rules file then tells Analytic Services to read the header information as a header record and not a data record. You can also specify which type of header information is in which header record.

  Header information defined in the data source takes precedence over header information defined in the rules file.

➤ To define a header in the rules file, see “Setting Headers in the Rules File” in the *Essbase Administration Services Online Help*.

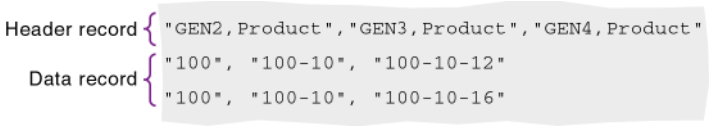
➤ To define a header in the data source, see “Setting Headers in the Data Source” in the *Essbase Administration Services Online Help*.

Data Source Headers

You can dynamically build dimensions by adding header information to the top record of the data source and by specifying the location of the header record in the rules file.

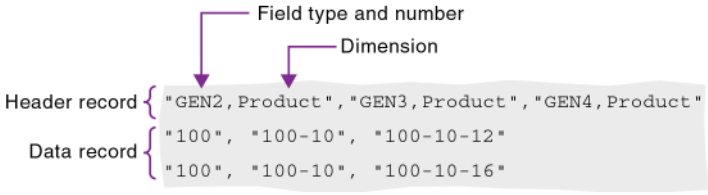
Figure 105 contains an example of a header record.

Figure 105: Header Record



The header record lists field definitions for each field. The field definition includes the field type, the field number, and the dimension name into which to load the fields. The format of a header record is illustrated in Figure 106:

Figure 106: Header Record with Three Field Definitions



If the file delimiter is a comma, enclose each field definition in quotation marks ("").

After you set the header information in the data source, you must specify the location of the header information in the rules file. If a rules file refers to header information in a data source, Analytic Services uses the information in the data source—rather than the information in the rules file—to determine field types and dimensions.

Valid Data Source Header Field Types

Valid field types must be in capital letters and are as follows:

- GEN, DUPGEN, and DUPGENALIAS
- LEVEL, DUPLEVEL, and DUPLEVELALIAS
- PARENT, CHILD
- PROPERTY
- ALIAS
- FORMULA
- CURNAME
- CURCAT
- UDA
- ATTRPARENT
- The name of an attribute dimension, such as CAFFEINATED

For each field type that you set, you must also enter a field number. When the field type is the name of an attribute dimension, the field number cannot be greater than 9. For a brief discussion and references to pertinent topics, see "Setting Field Type Information" on page 381.

Performing Operations on Fields

You can perform operations at the field level, for example, moving a field to a new position in the record.

This section contains the following sections:

Ignoring Fields

You can ignore all fields of a specified column of the data source. The fields still exist in the data source, but they are not loaded into the Analytic Services database.

If the data source contains fields that you do not want to load into the database, tell Analytic Services to ignore those fields. For example, the Sample Basic database has five standard dimensions: Year, Product, Market, Measures, and Scenario. If the data source has an extra field, such as Salesperson, that is not a member of any dimension, ignore the Salesperson field.

➤ To ignore all fields in a column, see "Ignoring Fields" in the *Essbase Administration Services Online Help*.

Ignoring Strings

You can ignore any field in the data source that matches a string called a *token*. When you ignore fields based on string values, the fields are ignored everywhere they appear in the data source, not just in a particular column. Consider, for example, a data source that is a computer generated report in text format. Special ASCII characters might be used to create horizontal lines between pages or boxes around headings. These special characters can be defined as tokens to be ignored.

➤ To ignore all instances of a string, see "Ignoring Fields Based on String Matches" in the *Essbase Administration Services Online Help*.

Arranging Fields

You can set the order of the fields in the rules file to be different from the order of the fields in the data source.The data source is unchanged. The following sections describe:

- “Creating Additional Text Fields” on page 398
- “Undoing Field Operations” on page 398

Note: To undo a single operation, select Edit > Undo. To undo one or more field operations, see “Undoing Field Operations” in the *Essbase Administration Services Online Help*.

Moving Fields

You can move fields to a different location using a rules file. For example, a field might be the first field in the data source, but you want to move it to be the third field during the data load or dimension build.

In some instances, moved fields may appear to merge. Merging may occur if the data file has a structure similar to the following:

```
1<tab>2<tab>3
1<tab>2<tab>(null)
```

If you move a field that contains empty cells and the moved field becomes the last field in the record, the field may merge with the field to its left.

To prevent merging, replace the empty cell with a delimiter.

➤ To move fields, see “Moving Fields” in the *Essbase Administration Services Online Help*.

Note: To undo a move, see “Undoing Field Operations” in the *Essbase Administration Services Online Help*.

Joining Fields

You can join multiple fields into one field. The new field is given the name of the first field in the join. For example, if you receive a data source with separate fields for product number (100) and product family (-10), you must join the fields (100-10) before you load them into the Sample Basic database.

Before you join fields, move the fields to join into the order in which you want to join them. If you do not know how to move fields, see “Moving Fields” in the *Essbase Administration Services Online Help*.

➤ To join fields, see “Joining Fields” in the *Essbase Administration Services Online Help*.

Note: To undo a join, see “Undoing Field Operations” in the *Essbase Administration Services Online Help*.

Creating a New Field by Joining Fields

You can join two or more fields by placing the joined fields into a new field. This procedure leaves the original fields intact. Creating a new field is useful if you need to concatenate fields of the data source to create a member.

For example, if you receive a data source with separate fields for product number (100) and product family (-10), you must join the fields (100-10) before you load them into the Sample Basic database. But suppose that you want the 100 and -10 fields to exist in the data source after the join; that is, you want the data source to contain three fields: 100, -10, and 100-10. To do this, create the new field using a join.

Before you join fields, move the fields to join into the order in which you want to join them. If you do not know how to move fields, see “Moving Fields” on page 396.

➤ To create a new field by joining existing fields, see “Creating a New Field Using Joins” in the *Essbase Administration Services Online Help*.

Note: To undo a creating using join operation, see “Undoing Field Operations” in the *Essbase Administration Services Online Help*.

Copying Fields

You can create a copy of a field while leaving the original field intact. For example, assume that, during a single dimension build, you want to define a multilevel attribute dimension and associate attributes with members of a base dimension. To accomplish this task, you need to copy some of the fields. For more information about attribute dimensions, see “Working with Multilevel Attribute Dimensions” on page 438.

➤ To copy a field, select one field and then create a new field using a join.

➤ To create a new field by joining existing fields, see "Creating a New Field Using Joins" in the *Essbase Administration Services Online Help*.

Note: To undo a copy, see "Undoing Field Operations" in the *Essbase Administration Services Online Help*.

Splitting Fields

You can split a field into two fields. For example, if a data source for the Sample Basic database has a field containing UPC100-10-1, you can split the UPC out of the field and ignore it. Then only 100-10-1, that is, the product number, is loaded. To ignore a field, see "Ignoring Fields" on page 395.

➤ To split a field, see "Splitting Fields" in the *Essbase Administration Services Online Help*.

Note: To undo a split, see "Undoing Field Operations" in the *Essbase Administration Services Online Help*.

Creating Additional Text Fields

You can create a text field between two existing fields. You might create a text field to insert text between fields that are to be joined. For example, if you have two fields, one containing 100 and one containing 10-1, you can insert a text field with a dash between the two fields and then join the three fields to create the 100-10-1 member of the Product dimension.

➤ To create a new field and populate it with text, see "Creating a New Field Using Text" in the *Essbase Administration Services Online Help*.

Note: To undo a field you created using text, see "Undoing Field Operations" in the *Essbase Administration Services Online Help*.

Undoing Field Operations

You can undo the last field operation that you performed, such as move, split, join, create using text, or create using join by using the Edit > Undo command. You can also undo field operations even if you have performed other actions. Undoing field operations is sequential; you must undo field operations from the last operation to the first operation.

➤ To undo one or more field operations, see “Undoing Field Operations” in the *Essbase Administration Services Online Help*.

Mapping Fields

This section applies to data load only. If you are performing a dimension build, skip this section.

You use a rules file to map data source fields to Analytic Services member names during a data load. You can map fields in a data source directly to fields in the Analytic Services database during a data load by specifying which field in the data source maps to which member or member combination in the Analytic Services database. The data source is not changed.

Note: When you open a SQL data source, the fields default to the SQL data source column names. If the SQL column names and the Analytic Services dimension names are the same, you do not have to map the column names.

➤ To map fields, see “Mapping Field Names” in the *Essbase Administration Services Online Help*.

Changing Field Names

To load a data source, you must specify how the fields of the data source map to the dimensions and members of the database. Rules files can translate fields of the data source so that the fields match member names each time the data source is loaded. This process does not change the data source. The rules file does the following:

- Maps member fields of the data source to dimensions and members of the database
- Maps data fields of the data source to member names or member combinations (such as Jan, Actual) of the database

This section contains the following sections that describe how to change field names in the data source to map members and data values to the database.

- "Converting Spaces to Underscores" on page 401
- "Adding Prefixes or Suffixes to Field Values" on page 401

Replacing Text Strings

You can replace text strings so that the fields map to Analytic Services member names during a data load or dimension build. The data source is not changed. For example, if the data source abbreviates New York to NY, you can have the rules file replace each NY with New York during the data load or the dimension build.

For instructions on how to replace an empty field with text, see "Replacing an Empty Field with Text" on page 400.

➤ To replace a text string, see "Replacing Field Names" in the *Essbase Administration Services Online Help*.

Replacing an Empty Field with Text

You may want to replace empty fields in a column with text. If, for example, empty fields in the column represent default values, you can insert the default values or insert `#MI` to represent missing values.

➤ To replace an empty field with text, see "Replacing an Empty Field with Text" in *Essbase Administration Services Online Help*.

Changing the Case of Fields

You can change the case of a field so the field maps to Analytic Services member names during a data load or dimension build. The data source is not changed. For example, if the data source capitalizes a field that is in lower case in the database, you could change the field to lower case; for example, JAN to jan.

➤ To change the case of values in a field, see "Changing Case of Fields" in the *Essbase Administration Services Online Help*.

Dropping Leading and Trailing Spaces

You can drop leading and trailing spaces from around fields of the data source. A field value containing leading or trailing spaces does not map to a member name, even if the name within the spaces is an exact match.

By default, Analytic Services drops leading and trailing spaces.

➤ To drop spaces around a field, see "Dropping Spaces Around Fields" in the *Essbase Administration Services Online Help*.

Converting Spaces to Underscores

You can convert spaces in fields of the data source to underscores to make the field values match the member names of the database.

➤ To change spaces to underscores, see "Converting Spaces to Underscores" in the *Essbase Administration Services Online Help*.

Adding Prefixes or Suffixes to Field Values

You can add prefixes and suffixes to each field value of the data source. For example, you can add 2002 as the prefix to all member names in the Year dimension.

➤ To prefix or suffix values to a field, see "Adding Prefixes and Suffixes" in the *Essbase Administration Services Online Help*.

Performing Operations on Data

This section applies to data load only. If you are performing a dimension build, skip this section.

You can perform operations on the data in a field, for example, moving a field to a new position in the record.

This section contains the following sections:

- “Scaling Data Values” on page 404
- “Flipping Field Signs” on page 404

Defining a Column as a Data Field

This section applies to data load only. If you are performing a dimension build, skip this section.

If each record in the data source contains a column for every dimension and one data column, you must define the data column as a data field. In Figure 107, for example, the column with the data values must be defined as a data field.

Figure 107: Data Field

```
Market, Product, Year, Measures, Scenario

Texas  100-10  Jan  Sales Actual 42
Texas  100-20  Jan  Sales Actual 82
Texas  100-10  Jan  Sales Actual 37
```

You can define only one field in a record as a data field.

➤ To define a data field, see “Defining a Column as a Data Field” in the *Essbase Administration Services Online Help*.

Adding to and Subtracting from Existing Values

This section is for data load only. If you are performing a dimension build, skip this section.

By default, Analytic Services overwrites the existing values of the database with the values of the data source, but you can determine how newly loaded data values affect existing data values.

You can use incoming data values to add to or subtract from existing database values. For example, if you load weekly values, you can add them to create monthly values in the database.

Using this option makes it more difficult to recover if the database crashes while loading data, although Analytic Services lists the number of the last row committed in the application log. For a discussion of the application log, see “Contents of the Application Log” on page 983.

To prevent difficult recoveries if you are adding to or subtracting from existing data values and the database shuts down abnormally, as a Database Transaction setting, set the Commit Row value as 0. This setting causes Analytic Services to view the entire load as a single transaction and to commit the data only when the load is complete. For more information, see "Understanding Isolation Levels" on page 1054.

- To add to existing data values, see "Adding to Data Values" in the *Essbase Administration Services Online Help*.
- To subtract from existing data values, see "Subtracting from Data Values" in the *Essbase Administration Services Online Help*.

Clearing Existing Data Values

This section is for data load only. If you are performing a dimension build, skip this section.

You can clear existing data values from the database before you load new values. By default, Analytic Services overwrites the existing values of the database with the new values of the data source. If you are adding and subtracting data values, however, Analytic Services adds or subtracts the new data values to and from the existing values.

Before adding or subtracting new values, make sure that the existing values are correct. Before loading the first set of values into the database, you must make sure that there is no existing value.

For example, assume that the Sales figures for January are calculated by adding the values for each week in January:

```
January Sales = Week 1 Sales + Week 2 Sales + Week 3 Sales + Week 4 Sales
```

When you load Week 1 Sales, clear the database value for January Monthly Sales. If there is an existing value, Analytic Services performs the following calculation:

```
January Sales = Existing Value + Week 1 Sales + Week 2 Sales + Week 3 Sales
+ Week 4 Sales
```

You can also clear data from fields that are not part of the data load. For example, if a data source contains data for January, February, and March and you want to load only the March data, you can clear the January and February data.

Note: If you are using transparent partitions, clear the values using the steps that you use to clear data from a local database.

➤ To clear existing values, see “Clearing Existing Data Values” in the *Essbase Administration Services Online Help*.

Scaling Data Values

This section is for data load only. If you are performing a dimension build, skip this section.

You can scale data values if the values of the data source are not in the same scale as the values of the database.

For example, assume the real value of sales was $5,460. If the Sales data source tracks the values in hundreds, the value is 54.6. If the Analytic Services database tracks the real value, you need to multiply the value coming in from the Sales data source (54.6) by 100 to have the value display correctly in the Analytic Services database (as 5460).

➤ To scale data values, see “Scaling Data Values” in the *Essbase Administration Services Online Help*.

Flipping Field Signs

This section is for data load only. If you are performing a dimension build, skip this section.

You can reverse or flip the value of a data field by flipping its sign. Sign flips are based on the UDAs (user-defined attributes) of the outline. When loading data into the accounts dimension, for example, you can specify that any record whose accounts member has a UDA of Expense change from a plus sign to a minus sign. See “Creating UDAs” on page 176 for more information on user-defined attributes.

➤ To reverse a field sign, see “Flipping Signs” in the *Essbase Administration Services Online Help*.

Chapter 19

Performing and Debugging Data Loads or Dimension Builds

This chapter describes how to load data or members from one or more external data sources to an Analytic Server. You can load data without updating the outline, you can update the outline without loading data, or you can load data and build dimensions simultaneously. For information about setting up data sources and rules files, see Chapter 16, "Understanding Data Loading and Dimension Building" and Chapter 17, "Creating Rules Files."

This chapter contains the following sections:

Prerequisites for Data Loads and Dimension Builds

Before you start to load data or build dimensions, make sure that you have the following items in place:

- An Analytic Services database.
- A connection to the appropriate Analytic Server.
- One or more valid data sources. For a comprehensive discussion of data sources, see "Data Sources" on page 356.

- If you are not using a rules file, a data source correctly formatted for free-form data loading, see "Data Sources That Do Not Need a Rules File" on page 365.
- If you are using a rules file, for a definition and discussion of rules files, see "Rules Files" on page 364.

Performing Data Loads or Dimension Builds

When you start to load data or build dimensions, you must first select one or more valid data sources that contain the data to load or dimensions to build. For a list of types of valid data sources, see "Supported Data Sources" on page 357. Make sure you are connected to the Analytic Server before you specify the data sources. For a comprehensive discussion of how to optimize a data load, see Chapter 53, "Optimizing Data Loads."

When you use Administration Services to perform a data load or dimension build for a block storage database, you can execute the load or build in the background so that you can continue working as the load or build processes. You can then check the status of the background process to see when the load or build has completed. For more information, see "Performing a Data Load or Dimension Build" in *Essbase Administration Services Online Help*.

Note: If you are loading data into a transparent partition, follow the same steps as for loading data into a local database.

➤ To load data or build dimensions, use any of the following methods:

Tool	Topic	Location
Administration Services	Performing a Data Load or Dimension Build	*Essbase Administration Services Online Help*
MaxL	For data loading: **import data** For dimension building: **import dimensions**	*Technical Reference*
ESSCMD	For data loading: IMPORT For dimension building: BUILDDIM	*Technical Reference*

Stopping Data Loads or Dimension Builds

You can stop a data load or dimension build before it completes. You should not stop a data load or dimension build unless you are very sure that stopping is necessary. If a data load or dimension build process is terminated, Analytic Services displays the file name as partially loaded.

If you initiate a data load or dimension build from a client and terminate the data load or dimension build from the server, it could take some time before the client responds to the termination request. Because Analytic Services reads the source file until all source data is read, the amount of time depends on the size of the file and the amount of source data that Analytic Services has processed. If the process is terminated from the machine that initiated it, the termination is immediate.

Note: If you are adding to or subtracting from data values during a data load to a block storage database, use the Committed Isolation Level setting, if possible. If the data load is terminated, this setting rolls the data load back to its previous state. For a description of the operation of each isolation level setting, see "Understanding Isolation Levels" on page 1054. If you stop a data load that is adding to or subtracting from data values, see "Recovering from an Analytic Server Crash" on page 413 to identify the recovery procedure.

➤ To stop a data load or dimension build before it completes, use any of the following methods:

Tool	Topic	Location
Administration Services	Disconnecting User Sessions and Requests	*Essbase Administration Services Online Help*
MaxL	**alter system kill request**	*Technical Reference*

Reviewing the Tips for Loading Data and Building Dimensions

This section lists tips for data loading and dimension building. It contains the following sections

Determining Where to Load Data

Skip this section if you are building dimensions or working with an aggregate storage database.

If you load data into a parent member, when you calculate the database, the consolidation of the children's data values can overwrite the parent data value. To prevent overwriting, be aware of the following:

- If possible, do not load data directly into a parent.
- If you must load data into a parent member, make sure that Analytic Services knows not to consolidate #MISSING values from the children of the parent into the parent, as directed in the following table:

➤ To set the consolidation, use any of the following methods:

Tool	Topic	Location
Administration Services	Aggregating Missing Values During Calculation	*Essbase Administration Services Online Help*
Calculation Script	SET AGGMISSG	*Technical Reference*
MaxL	**alter database**	*Technical Reference*
ESSCMD	SETDBSTATEITEM	*Technical Reference*

The methods in this table work only if the child values are empty (#MISSING). If the children have data values, the data values overwrite the data values of the parent. For a discussion of how Analytic Services calculates #MISSING values, see "Consolidating #MISSING Values" on page 1217.

Note: You cannot load data into Dynamic Calc, Dynamic Calc and Store, or attribute members. For example, if Year is a Dynamic Calc member, you cannot load data into it. Instead, load data into Qtr1, Qtr2, Qtr3, and Qtr4, which are not Dynamic Calc members.

Loading Data Using a Spreadsheet

Skip this section if you are building dimensions.

If you use a spreadsheet to load data, see the *Essbase Administration Services Online Help* and search for "spreadsheet" in the index.

Dealing with Missing Fields in a Data Source

Each record in the data source must have the same number of fields to perform a data load or dimension build. If fields are missing, the data load or dimension build processes incorrectly. For example, the file in Figure 108 is invalid, because there is no value under Apr. To fix the file, insert #MISSING or #MI into the missing field. For instructions, see "Replacing an Empty Field with Text" on page 400.

Figure 108: Missing Fields

```
Actual Ohio Sales Cola
Jan     Feb    Mar    Apr
10      15     20
```

Figure 109 is valid because `#MI` replaces the missing field.

Figure 109: Valid Missing Fields

```
Actual Ohio Sales Cola
Jan     Feb    Mar    Apr
10      15     20     #MI
```

If a rules file has extra blank fields, join the empty fields with the field next to them. For a brief discussion, see "Joining Fields" on page 396.

Loading a Subset of Records from a Data Source

You can load a subset of records in a data source during a data load or a dimension build. For example, you can load records 250 to 500 without loading the other records of the data source.

➤ To load a subset of records:

1. Using a text editing tool, number the records in the data source.
2. Set the rules file to ignore the column containing the record number.

   For a brief discussion, see "Ignoring Fields" on page 395.
3. Define a rejection criterion that rejects all records except those that you want to load.

   For example, reject all records for which the ignored column is less than 250 or greater than 500. For a brief discussion, see "Rejecting Records" on page 390.

   Note: You cannot reject more records than the error log can hold. By default, the limit is 1000, but you can change it by setting DATAERRORLIMIT in the `essbase.cfg` file. See the *Technical Reference* for more information.

Debugging Data Loads and Dimension Builds

If you try to load a data source into Analytic Server, but it does not load correctly, check the following:

- Are you connected to the appropriate application and database?
- Are you trying to load the correct data source?

If you can answer both of the above questions with a "yes," something is probably wrong. Use the following sections to determine what the problem is and to correct the problem.

- "Verifying That Analytic Server Is Available" on page 411
- "Verifying That the Data Source Is Available" on page 411
- "Checking Error Logs" on page 412
- "Recovering from an Analytic Server Crash" on page 413
- "Resolving Problems with Data Loaded Incorrectly" on page 413

- “Creating Rejection Criteria for End of File Markers” on page 415
- “Understanding How Analytic Services Processes a Rules File” on page 415
- “Understanding how Analytic Services Processes Invalid Fields During a Data Load” on page 417

When you correct the problems, you can reload the records that did not load by reloading the error log. For more information, see “Loading Dimension Build and Data Load Error Logs” on page 1018.

Verifying That Analytic Server Is Available

To help identify if the problem is with Analytic Services and not with the server or network, try to access the server without using Analytic Services. Check the following:

- Is the server machine running? Try to connect to it without using Analytic Services. If you cannot, check with your system administrator.
- Is Analytic Server running? Check with your Analytic Services administrator.
- Can the client machine connect to the server machine? Try to connect to the server machine from the client machine without using Analytic Services.

Verifying That the Data Source Is Available

If Analytic Services cannot open the data source that you want to load, check the following:

- Is the data source already open? The data source will already be open if a user is editing the data source. Analytic Services can load only data sources that are not locked by another user or application.
- Does the data source have the correct file extension? All text files must have a file extension of `.TXT`. All rules files must have a file extension of `.RUL`.
- Is the data source name and the path name correct? Check for misspellings.
- Is the data source in the specified location? Check to make sure that no one has moved or deleted the data source.

- If you are using a SQL data source, is the connection information (such as the user name, password, and database name) correct?
- If you are using a SQL data source, can you connect to the SQL data source without using Analytic Services?

Checking Error Logs

If a data load or dimension build fails, the error log can be a valuable debugging tool. See "Understanding and Viewing Dimension Build and Data Load Error Logs" on page 1016 in for more information about error logs.

If there is no error log, check the following:

- Did the person running the data load set up an error log? Click Help in the Data Load dialog box for information on setting up an error log. By default, when you use a rules file, Analytic Services creates an error log.
- Are you sure that the data source and Analytic Server are available? See "Verifying That Analytic Server Is Available" on page 411 and "Verifying That the Data Source Is Available" on page 411 for lists of items to check.
- Did the Analytic Server crash during the data load? If so, you probably received a time-out error on the client. If the server crashed, see "Recovering from an Analytic Server Crash" on page 413 to identify a recovery procedure.
- Check the application log. For a review of log information, see "Analytic Server and Application Logs" on page 979.

If the error log exists but is empty, Analytic Services does not think that an error occurred during loading. Check the following:

- Does the rules file contain selection or rejection criteria that rejected every record in the data source? See "Selecting Records" on page 390 and "Rejecting Records" on page 390 for a discussion of how to set selection and rejection criteria.
- Is the rules file correct? Does the rules file validate properly? See "Requirements for Valid Data Load Rules Files" on page 386 and "Requirements for Valid Dimension Build Rules Files" on page 387 for a discussion of causes of non validation.

Recovering from an Analytic Server Crash

If the server crashes while you are loading data, Analytic Services sends you a time-out error. The recovery procedures that you need to perform depend on the type of load you are performing and the Isolation Level setting:

- If you are overwriting the values of the data source, reload the data source when the server is running again.
- If you are adding to or subtracting from existing values in the data source and the Isolation Level transaction setting is Committed, reload the data source when the server is running again.
- If you are adding to or subtracting from existing values in the data source and the Isolation Level is Uncommitted, determine how much data Analytic Services loaded before the crash:
  a. Compare the values of the data source with the values of the database.
  b. If the values that you are adding to or subtracting from have not changed, reload the data source.
  c. If the values that you are adding to or subtracting from have changed, clear the values that loaded and reload the previous data sources. If, for example, you derive monthly sales figures by adding the sales figures for each week as they are loaded, clear the sales figures from the database and re-load the sales figures for each week up to the current week.

For a description of Isolation Level settings, see "Understanding Isolation Levels" on page 1054.

Resolving Problems with Data Loaded Incorrectly

If the data source loads without error, but the data in the database is wrong, check the following:

- Are you sure that you loaded the correct data source? If so, check the data source again to make sure that it contains the correct values.
- Are there any blank fields in the data source? You must insert #MI or #MISSING into a data field that has no value. Otherwise, the data source may not load correctly. To replace a blank field with #MI or #MISSING using a rules file, see "Replacing an Empty Field with Text" on page 400.

- Is the data source formatted correctly?
  - Are all ranges set up properly?
  - Is the data clean? For example, as it processes the data source, Analytic Services recognizes member names and knows the dimensions they belong to. If a data source record inadvertently includes a member from a dimension for which there is a member named in the header record, the new member name replaces the header record member for that dimension. Consider the following example data source:

    ```
    Jan     Actual     Texas     Sales
    "100-10"     51.7
    "100-20"     102.5
    "100-20"     335.0
    Florida      96.7
    "200-20"     276.0
    "200-20"     113.1
    "200-10"     167.0
    ```

    Analytic Services recognizes Florida to be a member of the Market dimension. The values in the last four records are interpreted as Florida values instead of Texas values.
- Are there any implicitly shared members that you were unaware of? Implicit shares happen when a parent and child share the same data value. This situation occurs if a parent has only one child or only one child rolls up into the parent. For a definition and discussion of implied sharing, see "Understanding Implied Sharing" on page 168.
- Did you add incoming data to existing data instead of replacing incoming data with existing data? For a discussion of the adding and subtracting process, see "Adding to and Subtracting from Existing Values" on page 402.
- Have you selected or rejected any records that you did not intend to select or reject? For a brief discussion of selecting and rejecting, see "Selecting Records" on page 390 and "Rejecting Records" on page 390.
- If the sign is reversed (for example, a minus sign instead of a plus sign), did you perform any sign flips on UDAs (user-defined attributes)? For a discussion of the sign flipping process, see "Flipping Field Signs" on page 404.
- Did you clear data combinations that you did not intend to clear? For a discussion of the process of clearing data, see "Clearing Existing Data Values" on page 403.

- Did you scale the incoming values incorrectly? For examples of scaling data, see "Scaling Data Values" on page 404.
- Are all member and alias names less than 79 characters long?

Note: You can check data by exporting it, by running a report on it, or by using a spreadsheet. If doing exports and reports, see Chapter 32, "Developing Report Scripts" and Appendix D, "Using ESSCMD." If using a spreadsheet, see the *Essbase Spreadsheet Add-in User's Guide*.

Creating Rejection Criteria for End of File Markers

A SQL data source may have an end of file marker made up of special characters that cause a data load or dimension build to fail. To fix this problem, define a rejection criterion to reject the problem record.

1. Find the end of file marker in the SQL data source.
2. Determine how to search for it using the Analytic Services search command.

   This task may be difficult as the end of file marker may be composed of one or more special characters. To ignore all instances of a string, see "Ignoring Fields Based on String Matches" in the *Essbase Administration Services Online Help*.
3. Define a rejection criterion that rejects the end of file marker.

   See "Rejecting Records" in the *Essbase Administration Services Online Help*.

Understanding How Analytic Services Processes a Rules File

Sometimes, you can track down problems with dimension builds by understanding how Analytic Services initializes the rules file and processes the data source.

Analytic Services performs the following steps to initialize a rules file:

1. Validates the rules file against the associated outline.
2. Validates the dimensions. This process includes ensuring that the build method and field types are compatible and that each dimension name is unique. Member names must be either unique or shared.
3. Adds new dimensions defined in the rules file to the outline.
4. Reads header records specified in the data source.

Then Analytic Services performs the following operations on each record of the data source during a data load or dimension build:

1. Sets the file delimiters for all records.
2. Applies field operations to the data in the order that the operations are defined in the rules file. Field operations include joins, moves, splits, and creating fields using text and joins. To see the order in which field operations are defined in the rules file, see "Undoing Field Operations" on page 398. The dialog box displayed lists all the field operations in order.
3. Analytic Services applies all properties for each field, applying all properties to field1 before proceeding to field2. Analytic Services applies field properties in the following order:
   a. Ignores fields set to be ignored during data load.
   b. Ignores fields set to be ignored during dimension build.
   c. Flags the data field.
   d. Applies field names.
   e. Applies field generations.
   f. Performs all replaces in the order that they are defined in the rules file.
   g. Drops leading and trailing spaces.
   h. Converts spaces to underscores.
   i. Applies suffix and prefix operations.
   j. Scales data values.
   k. Converts text to lowercase.
   l. Converts text to uppercase.
4. Adds members or member information, or both, to the outline.
5. If you chose to skip lines, Analytic Services skips the number of lines that you specified; otherwise, Analytic Services proceeds to the first record.
6. Analytic Services performs selection or rejection criteria in the order that the criteria are defined in the rules file. Analytic Services loads or rejects individual records of the data source based on the specified criteria.

Understanding how Analytic Services Processes Invalid Fields During a Data Load

The following sections describe how Analytic Services processes invalid fields during a data load.

Missing Dimension or Member Fields

If you are using a rules file for the data load, skip this section. It applies only to data loaded without a rules file.

In a free-form data load, if a dimension or member field is missing, Analytic Services uses the value that it used previously for that dimension or member field. If there is no previous value, Analytic Services aborts the data load.

For example, when you load Figure 110 into the Sample Basic database, Analytic Services maps the Ohio member field into the Market dimension for all records, including the records that have Root Beer and Diet Cola in the Product dimension.

Figure 110: Valid Missing Members

```
Jan Sales Actual Ohio
Cola          25
"Root Beer"   50
"Diet Cola"   19
```

Analytic Services stops the data load if no prior record contains a value for the missing member field. If you try to load Figure 111 into the Sample Basic database, for example, the data load stops, because the Market dimension (Ohio, in Figure 110) is not specified.

Figure 111: Invalid Missing Members

```
Jan Sales Actual
           Cola          25
           "Root Beer"   50
           "Diet Cola"   19
```

For information on restarting the load, see "Loading Dimension Build and Data Load Error Logs" on page 1018.

Unknown Member Fields

If you are performing a data load and Analytic Services encounters an unknown member name, Analytic Services rejects the entire record. If there is a prior record with a member name for the missing member field, Analytic Services continues to the next record. If there is no prior record, the data load stops. For example, when you load Figure 112 into the Sample Basic database, Analytic Services rejects the record containing Ginger Ale because it is not a valid member name. Analytic Services loads the records containing Cola, Root Beer, and Cream Soda. If Ginger Ale were in the first record, however, the data load would stop.

Figure 112: Unknown Members

```
Jan, Sales, Actual
Ohio    Cola          2
        "Root Beer"   12
        "Ginger Ale"  15
        "Cream Soda"  11
```

Note: If you are performing a dimension build, you can add the new member to the database. See "Performing Data Loads or Dimension Builds" on page 406.

For information on restarting the load, see "Loading Dimension Build and Data Load Error Logs" on page 1018.

Invalid Data Fields

If you are performing a data load, when Analytic Services encounters an invalid data field, it stops the data load. Analytic Services loads all fields read before the invalid field into the database, resulting in a partial load of the data. In the following file, for example, Analytic Services stops the data load when it encounters the 15- data value. Analytic Services loads the Jan and Feb Sales records, but not the Mar and Apr Sales records.

Figure 113: Invalid Data Field

```
East Cola   Actual
Sales       Jan     $10
            Feb     $21
            Mar     $15-
            Apr     $16
```

For information on continuing the load, see "Loading Dimension Build and Data Load Error Logs" on page 1018.

Chapter 20

Understanding Advanced Dimension Building Concepts

This chapter discusses dimension building.

Understanding Build Methods

The build method that you select determines the algorithm that Analytic Services uses to add, change, or remove dimensions, members, and aliases in the outline. The kind of build method that you select depends on the type of data in the data source.

The following table provides guidelines to help you select the appropriate build method for the data source:

Table 22: Build Method Guidelines

Type of Data in Each Record	Examples	Desired Operation	Build Method	Field Type Information
Top-down data: Each record specifies the parent's name, the child's name, the children of that child, and so forth.	Year, Quarter, Month	Modify the properties of existing dimensions and members	Generation references	The generation number for each field.
Bottom-up data: Each record specifies the name of the member, the name of its parent, the name of its parent's parent, and so forth.	Month, Quarter, Year	• Create shared members that roll up into different generations • Modify the properties of existing dimensions and members	Level references	The level number for each field.
Parent followed by its child: Each record specifies the name of the parent and the name of the new child member, in that order, although they can specify other information as well.	Cola, Diet Cola	• Create shared members that roll up into different generations • Share non-leaf members • Modify properties of existing dimensions and members	Parent-child references	Whether a field is parent or child. The field number is 0.

Table 22: Build Method Guidelines (Continued)

Type of Data in Each Record	Examples	Desired Operation	Build Method	Field Type Information
A list of new members: Each data source lists new members; the data source does not specify where in the outline the members belong. Analytic Services provides algorithms that determine where to add these members.	Jan, Feb, Mar, April	Add all members as children of an existing parent (possibly a "dummy" parent)	Add as child of the specified parent	
	800-10, 800-20	Add all members at the end of the dimension	Add as sibling at the lowest level	
	800-10, 800-20	Add each new member to the dimension that contains similar members	Add as sibling to a member with a matching string	
A list of base dimension members and their attributes.	Cola 16oz Can, Root Beer 14oz Bottle	Add members to an attribute dimension and associate the added members with the appropriate members of the base dimension	Generation, level, or parent-child references, depending on the organization of the source data	The number for each field. The number is either the generation or level number of the associated member of the base dimension or zero.

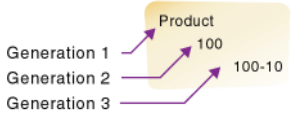
Using Generation References

Top-down data sources are organized left to right from the highest level to the lowest level. Each record begins with the most general information and progresses to the most specific information. The name of the new member is at the end of the record. When using a top-down data source, use the generation references build method. In the rules file, specify the generation number and the field type of each field of the data source.

Analytic Services numbers members within a dimension according to the hierarchical position of the member within the dimension. The numbers are called *generation references*. A dimension is always generation 1. All members at the same branch in a dimension are called a *generation*. Generations are numbered top-down according to their position relative to the dimension, that is, relative to dimension 1.

For example, as illustrated in Figure 114, the Product dimension in the Sample Basic database is generation 1. Product has a 100 member, which is generation 2. 100 has members, such as 100-10, which are generation 3. To use the generation references build method, you must specify the generation reference number in the rules file.

Figure 114: Generations



The top half of Figure 115 shows a top-down data source `GENREF.TXT`. The data source is used to build the Product dimension. The bottom half of Figure 115 shows the rules file for the data source, `GENREF.RUL`. The rules file specifies the generation number for each field in the data source. For information on setting field types and references to pertinent topics, see "Setting Field Type Information" on page 381.

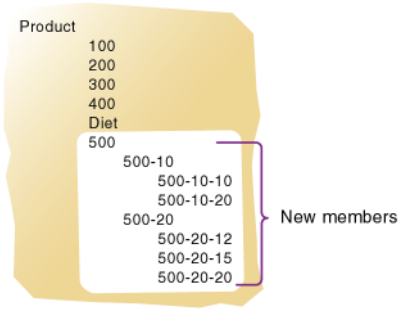
Figure 115: Rules File for Generation Build

```
1    500■500-10■500-10-10
2    500■500-10■500-10-20
3    500■500-20■500-20-12
4    500■500-20■500-20-15
5    500■500-20■500-20-20
```

	GEN2,Product	GEN3,Product	GEN4,Product
1	500	500-10	500-10-10
2	500	500-10	500-10-20
3	500	500-20	500-20-12
4	500	500-20	500-20-15
5	500	500-20	500-20-20

Figure 116 shows the tree that Analytic Services builds from this data source and rules file:

Figure 116: Generation References



Dealing with Empty Fields

When you use the generation references build method, you can choose to use null processing. Null processing specifies what actions Analytic Services takes when it encounters empty fields, also know as null fields, in the data source.

If null processing is not enabled, Analytic Services rejects all records with null values and writes an error to the error log.

If null processing is enabled, Analytic Services processes nulls as follows:

- If the null occurs where Analytic Services expects a GENERATION field, Analytic Services promotes the next GENERATION field to replace the missing field. In Figure 117, for example, there is no field in the GEN3,Products column. When Analytic Services reads the following record, it promotes the GEN4 field (100-10a) to GEN3.

  Figure 117: Missing Field in a Generation References Data Source

  ```
  GEN2,Products   GEN3,Products   GEN4,Products
  100                            100-10a
  ```

- If a null occurs directly before a secondary field, Analytic Services ignores the secondary field. Secondary field types are alias, property, formula, duplicate generation, duplicate generation alias, currency name, currency category, attribute parent, UDA, and name of an attribute dimension. In Figure 118, for example, there is no field in the GEN2, Products or the ALIAS2,Products

column. When Analytic Services reads the following record, it ignores the ALIAS2 field and promotes the GEN3 field (100-10) to GEN2 and the GEN4 field (100-10a) to GEN3.

Figure 118: Missing Secondary Field in a Generation References Data Source

```
GEN2,Products  ALIAS2,Products  GEN3,Products  GEN4,Products
               Cola             100-10         100-10a
```

- If the null occurs where Analytic Services expects a secondary field, Analytic Services ignores the secondary null field and continues loading. In Figure 119, for example, there is no field in the ALIAS2, Products column. When Analytic Services reads the following record, it ignores the ALIAS2 field.

  Figure 119: Missing Secondary Field in a Generation References Data Source

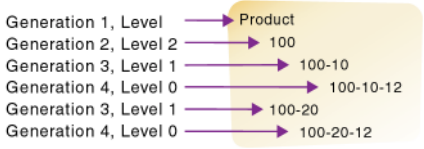
```
GEN2,Products  ALIAS2,Products  GEN3,Products  GEN4,Products
100                             100-10         100-10a
```

Using Level References

In a bottom-up data source, each record defines a single member of a dimension. The definition begins with the most specific information about the member and provides progressively more general information. A typical record specifies the name of the new member, then the name of its parent, then its parent's parent, and so forth.

Levels are defined from a bottom-up hierarchical structure. In the outline in Figure 120, for example, the lowest level members are at the bottoms of the branches of the Product dimension.

Figure 120: Generation and Level Numbers



To build the outline in Figure 120, you can use the bottom-up data source shown in Figure 121.

Figure 121: Bottom-up Data Source

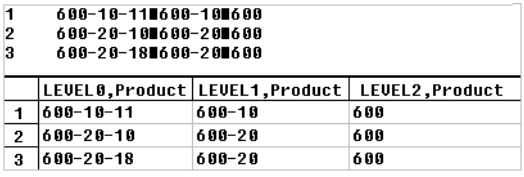
```
100-10-12 100-10 100
100-20-12 100-20 100
```

In a level reference build, the lowest level members are sequenced left to right. Level 0 members are in the first field, level 1 members are in the second field, and so on. This organization is the opposite of how data is presented for generation references (top-down).

The rules file in Figure 122 uses the level reference build method to add members to the Product dimension of the Sample Basic database. The first column of the data source contains new members (600-10-11, 600-20-10, and 600-20-18). The second column contains the parents of the new members (600-10 and 600-20), and the third column contains parents of the parents (600).

The rules file specifies the level number and the field type for each field of the data source. For more information on setting field types and references to pertinent topics, see "Setting Field Type Information" on page 381. To build the tree in Figure 123, for example, use Figure 122 to set up the data source, `LEVEL.TXT`, and the rules file, `LEVEL.RUL`.

Figure 122: Rules File for Level Build

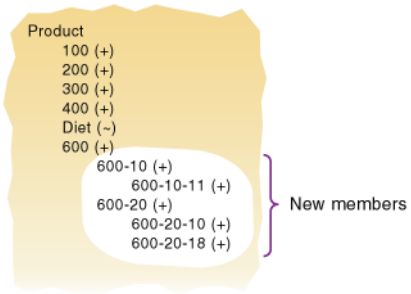


```
1    600-10-11■600-10■600
2    600-20-10■600-20■600
3    600-20-18■600-20■600
```

	LEVEL0,Product	LEVEL1,Product	LEVEL2,Product
1	600-10-11	600-10	600
2	600-20-10	600-20	600
3	600-20-18	600-20	600

Figure 123 shows the tree that Analytic Services builds from the data source and rules file of Figure 122.

Figure 123: Levels



Dealing with Empty Fields

When you use the level references build method, you can choose to use null processing. Null processing specifies what actions Analytic Services takes when it encounters empty fields, also know as null fields, in the data source.

If null processing is not enabled, Analytic Services rejects all records with null values and writes an error to the error log.

If null processing is enabled, Analytic Services processes nulls as follows:

- If a null occurs where Analytic Services expects a LEVEL field, Analytic Services promotes the next LEVEL field to replace the missing field. In Figure 124, for example, there is no field in the LEVEL0, Products column. When Analytic Services reads the following record, it promotes the LEVEL1 field (100-10) to LEVEL0 and the LEVEL2 field (100) to LEVEL1.

  Figure 124: Missing Field in a Level References Data Source

  ```
  LEVEL0,Products  LEVEL1,Products  LEVEL2,Products
                   100-10           100
  ```

- If a null occurs directly before a secondary field, Analytic Services ignores the secondary field. Secondary field options are alias, property, formula, duplicate level, duplicate level alias, currency name, currency category, attribute parent, UDA, and a name of an attribute dimension. In Figure 125, for example, there

is no field in the LEVEL0, Products column. When Analytic Services reads the following record, it ignores the ALIAS0 field and promotes the LEVEL1 field (100-10) to LEVEL0 and the LEVEL2 field (100) to LEVEL1.

Figure 125: Missing Secondary Field in a Level References Data Source

```
LEVEL0,Products  ALIAS0,Products  LEVEL1,Products  LEVEL2,Products
                 Cola             100-10           100
```

- If a null occurs where Analytic Services expects a secondary field, Analytic Services ignores the secondary null field and continues loading. In Figure 119, for example, there is no field in the ALIAS0, Products column. When Analytic Services reads the following record, it ignores the ALIAS0 field.

Figure 126: Missing Secondary Field in a Level References Data Source

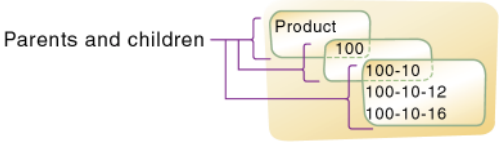
```
LEVEL0,Products  ALIAS0,Products  LEVEL1,Products  LEVEL2,Products
100-10a                           100-10           100
```

Using Parent-Child References

Use the parent-child references build method when every record of the data source specifies the name of a new member and the name of the parent to which you want to add the new member.

Members in a database exist in a parent-child relationship to one another. Figure 127 shows part of the Product dimension with its parent and children relationships identified.

Figure 127: Parents and Children



A parent-child data source must contain at least two columns: a parent column and a child column, in that order. The data source can include columns with other information (for example, the alias, the attributes or the properties of the new member). A record within a parent-child data source *cannot* specify more than one parent or more than one child and cannot reverse the order of the parent and child columns.

In a parent-child build, the rules file specifies which column is the parent and which column is the child. For general information on setting field types and references to pertinent topics, see "Setting Field Type Information" on page 381. For example, the top half of Figure 128 shows a data source, `PARCHIL.TXT`, in which each record specifies the name of a parent and the name of its child, in that order. The bottom half of the figure shows the rules file, `PARCHIL.RUL`, that specifies which column is the parent and which column is the child. In addition to identifying parent and child fields, this example associates aliases with the child field.

Figure 128: Rules Files for Parent-Child Build

```
1    200■200-10■Old Fashioned
2    200■200-20■Diet Root Beer
3    200■200-30■Sasparilla
4    200■200-40■Birch Beer
5    200■200-50■With Caffeine
```

	PARENT0,Product	CHILD0,Product	ALIAS0,Product
1	200	200-10	Old Fashioned
2	200	200-20	Diet Root Beer
3	200	200-30	Sasparilla
4	200	200-40	Birch Beer
5	200	200-50	With Caffeine

Figure 129 shows the tree that Analytic Services builds from this data source and rules file.

Figure 129: Parents and Children

Adding a List of New Members

If a data source consists of a list of new members and does not specify the ancestors of the new members, Analytic Services must decide where in the outline to add the new members. Analytic Services provides the following three build methods for this type of data source.

- Add each new member as a sibling of the existing member whose text most closely matches its own. For a discussion of the process and an example, see "Adding Members Based upon String Matches" on page 429.

- Add each new member as a sibling of the lowest-level existing member. For a discussion of the process and an example, see “Adding Members as Siblings of the Lowest Level” on page 431.
- Add all new members as children of a specified parent (generally a “dummy” parent). For a discussion of the process and an example, see “Adding Members to a Specified Parent” on page 432.

Note: Analytic Services does not support concurrent attribute association with the Add as build methods.

After Analytic Services adds all new members to the outline, it may be necessary to move the new members into their correct positions using Outline Editor. For a brief discussion and references to pertinent topics, see “Positioning Dimensions and Members” on page 147.

Adding Members Based upon String Matches

You can add new members from a data source to an existing dimension by matching strings with existing members. When Analytic Services encounters a new member in a data source, it scans the outline for a member name with similar text. Analytic Services then adds the new member as a sibling of the member with the closest string match.

For example, the data source in Figure 130, `SIBSTR.TXT`, contains two new members to add to the Product dimension in the Sample Basic database, 100-11 and 200-22. The new members are similar to strings in the Product dimension in that they contain 3 digits, 1 dash, and 2 digits.

To add the example members to the database, set the following values in the rules file:

In the rules file	Perform the following task	For brief discussions and references to pertinent topics
Select field 1 (Product).	• Do not select a field type for the field. • Set the dimension for the field to Product. Field 1 is displayed as Product, as shown in Figure 129.	See "Setting Field Type Information" on page 381.
Select field 2 through field 6.	Ignore the fields.	See "Ignoring Fields" on page 395.
Select the Product dimension.	Select the "Add as sibling of matching string" build method.	See "Selecting a Build Method" on page 378.

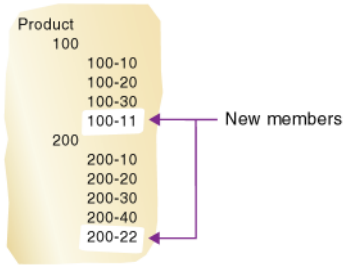
Figure 130: Rules File Fields Set to Add Members as Siblings with String Matches

```
1    100-11■Texas■Sales■100■120■100
2    200-22■Texas■Sales■111■154■180
```

	Product	field 2	field 3	field 4	field 5	field 6
1	100-11	Texas	Sales	100	120	100
2	200-22	Texas	Sales	111	154	180

Figure 131 shows the tree that Analytic Services builds from this data source and rules file.

Figure 131: Tree for Adding Members as Siblings with String Matches



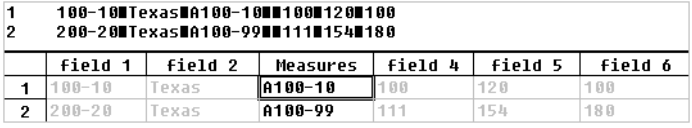
Adding Members as Siblings of the Lowest Level

You can add new members from a data source as siblings of members that reside at the lowest level of a dimension, that is, at the leaf branch. When Analytic Services encounters a new member in a data source, it scans the outline for the leaf branch of members. Analytic Services adds the new member as a sibling of these members.

Note: If the outline contains more than one group of members at this level, Analytic Services adds the new member to the first group of members that it encounters.

For example, the data source, `SIBLOW.TXT`, and the rules file, `SIBLOW.RUL`, in Figure 132 contain new members (A100-10 and A100-99) to add to the Measures dimension of the Sample Basic database.

Figure 132: Rules File Fields Set to Add Members as Siblings of the Lowest Level



To add the example members dynamically to the database, set the following values in the rules file:

In the rules file	Perform the following task	For brief discussions and references to pertinent topics
Select field 3 (Measures).	• Do not select a field type for the field. • Set the dimension for the field to Measures. Field 3 is displayed as Measures, as shown in Figure 132.	See "Setting Field Type Information" on page 381.
Select fields 1, 2, 4, 5, 6.	Ignore the fields.	See "Ignoring Fields" on page 395.
Select the Measures dimension.	Select the "Add as sibling of lowest level" build method.	See "Selecting a Build Method" on page 378.

Figure 133 shows the tree that Analytic Services builds from this data source and rules file.

Figure 133: Tree for Adding Members as Siblings of the Lowest Level

Adding Members to a Specified Parent

You can add all new members as children of a specified parent, generally a "dummy" parent. After Analytic Services adds all new members to the outline, review the added members and move or delete them in Outline Editor.

When Analytic Services encounters a new member in the data source, it adds the new member as a child of the parent that you define. The parent must be part of the outline before you start the dimension build.

For example, the data source in Figure 134, `SIBPAR.TXT`, contains two new members, 600-54 and 780-22, for the Product dimension (field 1). Assume that you previously added a member called NewProducts under the Products dimension.

Figure 134: Rules File Fields Set to Add Members as a Child of a Specified Parent

```
1    600-54■Texas■Sales■100■120■100
2    780-22■Texas■Sales■111■154■180
```

	Product	field 2	field 3	field 4	field 5	field 6
1	600-54	Texas	Sales	100	120	100
2	780-22	Texas	Sales	111	154	180

To add the example members to the database under the NewProducts member, set the following values in the rules file:

In the rules file	Perform the following task	For brief discussions and references to pertinent topics
Select field 1 (Product).	• Do not select a field type for the field. • Set the dimension for the field to Product. Field 1 is displayed as Product, as shown in Figure 134.	See "Setting Field Type Information" on page 381.
Select fields 2 through 6.	Ignore the fields.	See "Ignoring Fields" on page 395.
Select the Product dimension.	Select the "Add as child of" build method.	See "Selecting a Build Method" on page 378. Type **NewProducts** in the Add as Child of text box.

Figure 135 shows the tree that Analytic Services builds from this data source and rules file.

Figure 135: Tree for Adding Members as a Child of a Specified Parent

Building Attribute Dimensions and Associating Attributes

When a data source contains attribute information, you must use one or more rules files to build attribute dimensions and to associate attributes with members of their base dimensions.

You can use rules files to build attribute dimensions dynamically, to add and delete members, and to establish or change attribute associations.

Working with attributes involves the three following operations:

- If the base dimension does not exist, you must build it.
- You must build the attribute dimension.
- You must associate members of the base dimension with members of the attribute dimension.

You can use any of three approaches to perform these operations:

- Build both the base and attribute dimensions and perform the associations all at once. When you use an all-at-once approach, you use a single rules file to build the base dimension and one or more attribute dimensions and to associate the each attribute with the appropriate member of the base dimension. Because this approach uses a single rules file, it can be the most convenient. Use this approach if the base dimension does not exist and each source data record contains all attribute information for each member of the base dimension.
- Build the attribute dimension and perform the associations in one rules file. Assuming that the base dimension is built in a separate step or that the base dimension already exists, you can build an attribute dimension and associate the attributes with the members of the base dimension in a single step. You need only to define the attribute associations in the rules file. For a brief description of this process, see “Associating Attributes” on page 436.
- Build the attribute dimension and then perform the associations using separate rules files. Assuming that the base dimension is built in a separate step or that the base dimension already exists, you can build an attribute dimension and associate the attributes with the members of the base dimension in separate steps. Build the attribute dimension, and then associate the attribute members

with members of the base dimension. You must use this approach when you build numeric attribute dimensions that are multilevel or that have members that represent different-sized ranges.

The following sections describe how to build attribute dimensions:

Building Attribute Dimensions

Before you build any attribute dimensions in a database, you must define the attribute member name formats for the outline. For a comprehensive discussion of assigning attribute member names, see "Setting Member Names in Attribute Dimensions" on page 196.

You can build attribute dimensions in either of the following two ways:

- The same way that you build standard dimensions, as described in "Process for Data Loading and Dimension Building" on page 356.
- At the same time as you associate attributes with members of the base dimension, as described in "Associating Attributes" on page 436.

Analytic Services does not support concurrent attribute association with the Add as build methods.

When you define the rules file for building attribute dimensions, be sure to specify the base dimension and the name of the attribute dimension file.

Associating Attributes

Whether you build the attribute dimension and associate the attribute members with the members of the base dimension in one step or in separate steps, define the fields as described in this section.

Note: If you are working with a multilevel attribute dimension or with an attribute dimension of the type numeric, Boolean, or date, the rules file requires an additional field. For a complete example of a multilevel situation, see "Working with Multilevel Attribute Dimensions" on page 438.

Every record of the source data must include at least two columns, one for the member of the base dimension and one for the attribute value of the base dimension member. In the same source data record you can include additional columns for other attributes that you want to associate with the member of the base dimension. You must position the field for the member of the base dimension before any of the fields for the members of the attribute dimension.

Define the field type for the attribute dimension member as the name of the attribute dimension, use the generation or level number of the associated member of the base dimension, and specify the base dimension name. For example, as shown in the `ATTRPROD.RUL` file in Figure 136, the field definition Ounces3,Product specifies that the field contains members of the Ounces attribute dimension. Each member of this field is associated with the data field that is defined as the generation 3 member of the base dimension Product. Based on this field definition, Analytic Services associates the attribute 64 with the 500-10 member.

Figure 136: Rules File for Associating Attributes

```
1    500■500-10■64■True
2    500■500-20■64■False
```

	GEN2,Product	GEN3,Product	Ounces3,Product	Caffeinated3,Product
1	500	500-10	64	True
2	500	500-20	64	False

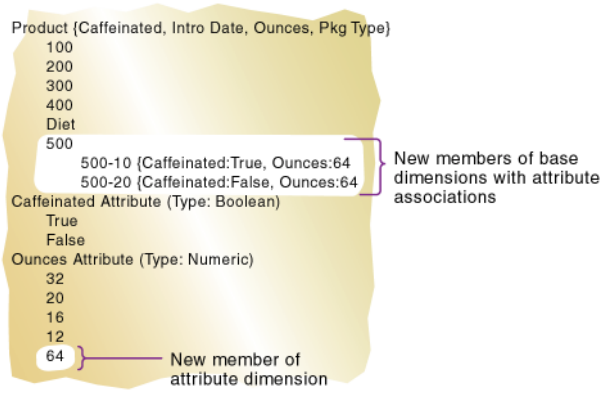
You can have Analytic Services use the attribute columns to build the members of the attribute dimensions. In Data Prep Editor, in the Dimension Build Settings tab of the Dimension Build Settings dialog box, for the base dimension, clear the Do Not Create Mbrs option. For more information, see "Setting Member Properties" in the *Essbase Administration Services Online Help*.

When you are working with numeric ranges, you may need to build attribute dimensions and perform associations in separate steps. For a discussion and example of using separate steps, see "Working with Numeric Ranges" on page 441.

The Caffeinated3,Product field in the example in Figure 136 shows how to associate attributes from additional single-level attribute dimensions. Because the base dimension is already specified, you need only to define an additional field for each attribute that you want to associate with the member of the base dimension.

The file in Figure 136 associates attributes as shown in the outline in Figure 137. The members 500, 500-10, and 500-20 are new members of the base dimension, Product. The member 64 is a new member of the Ounces attribute dimension.

Figure 137: Associating Attributes



Updating Attribute Associations

You can also use the rules file shown in Figure 136 to change attribute associations. Make sure that you allow association changes. In Data Prep Editor, in the Dimension Build Settings tab of the Dimension Build Settings dialog box, check "Allow Association Chgs" for the base dimension. For more information, see "Setting Member Properties" in the *Essbase Administration Services Online Help*.

Working with Multilevel Attribute Dimensions

Multilevel, numeric, Boolean, and date attribute dimensions can have duplicate level 0 members. For example, associated with a Product dimension you can have a Size attribute dimension with two levels. Level 1 categorizes sizes by men or by women. The level 0 members (attributes) are the actual sizes. You can have a member named 8 under Women and member named 8 under Men.

When an attribute is part of a multilevel numeric, Boolean, or date attribute dimension, the source data must include columns for all generations or levels of the attribute dimension. In the rules file, you must make copies of all fields that comprise the levels of the attribute dimension. Define the first set of attribute fields to build the attribute dimension. Define the second set of attribute fields to associate the attributes with the appropriate base dimension members. To ensure association with the correct attribute, indicate the parent field for the attribute field by making a copy of the parent field and setting the copy of the parent field as the field type Attribute Parent.

The position of the fields in the rules file is important.

- Place the copied attribute dimension field or fields that define the association immediately to the right of the field for the members of the base dimension.
- For a multilevel attribute dimension, place the attribute parent field immediately to the left of the field that is the child of the attribute parent.

The following steps describe how to define the fields in the rules file to build a multilevel attribute dimension and associate its members with members of its base dimension. This example uses the level references build method.

Note: For brief discussions of and references to topics pertinent to the following steps, see "Setting Field Type Information" on page 381, "Copying Fields" on page 397, and "Moving Fields" on page 396.

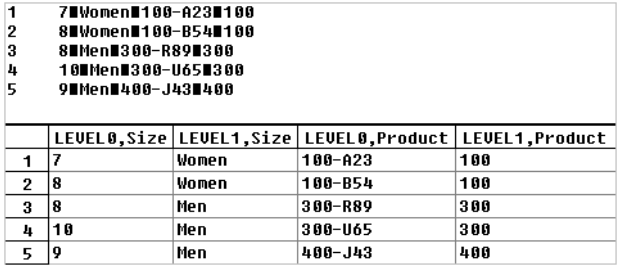
1. In the rules file, in field 1 and field 2, define the attribute dimension fields in the same way that you define standard dimensions; specify type (level or generation), number, and dimension name.

   Analytic Services uses the field1 and field2 to build the attribute dimension.

2. Define the fields for building the base dimension.

   In the following example, you are defining the level 0 and level 1 fields for the Product dimension. Figure 138 shows the fields of the rules file at this stage.

   Figure 138: Defining Multilevel Attribute Dimensions Before Adding the Association Fields



```
1    7■Women■100-A23■100
2    8■Women■100-B54■100
3    8■Men■300-R89■300
4    10■Men■300-U65■300
5    9■Men■400-J43■400
```

	LEVEL0,Size	LEVEL1,Size	LEVEL0,Product	LEVEL1,Product
1	7	Women	100-A23	100
2	8	Women	100-B54	100
3	8	Men	300-R89	300
4	10	Men	300-U65	300
5	9	Men	400-J43	400

3. To define the association, make a copy of the field that contains the level 0 attribute.

   In the current example, make a copy of field 1.

   a. Use the attribute dimension name as the field type and specify the generation or level number of the member of the base dimension with which Analytic Services associates the attribute; for example, Size0.

   b. Specify the base dimension; for example, Product.

   c. Move the new field immediately to the right of the field for the base dimension with which Analytic Services associates the attribute.

      In the current example, move the new field to the right of the field Level0, Product.

4. Make a copy of the field containing the parent of the attribute field.

   In the current example, make a copy of field 2.

   a. Set the field type of the new field as Attribute Parent and specify the generation or level number of the base member with which you want Analytic Services to associate the attribute; for example, ATTRPARENT0.

   b. Specify the attribute dimension; for example, Size.

   c. Move the ATTRPARENT field immediately to the left of the attribute association field that you created in step 3.

As shown in Figure 139, the rules file now contains the field definitions to build the attribute dimension Size and to associate the members of Size with the appropriate members of the base dimension Product.

Figure 139: Source Data and Rules File for Building a Multilevel Attribute Dimension

```
1    7■Women■100-A23■100
2    8■Women■100-B54■100
3    8■Men■300-R89■300
4    10■Men■300-U65■300
5    9■Men■400-J43■400
```

	LEVEL0,Size	LEVEL1,Size	LEVEL0,Product	ATTRPARENT0,Size	Size0,Product	LEVEL1,Product
1	7	Women	100-A23	Women	7	100
2	8	Women	100-B54	Women	8	100
3	8	Men	300-R89	Men	8	300
4	10	Men	300-U65	Men	10	300
5	9	Men	400-J43	Men	9	400

When you run a dimension build with the data shown in Figure 139, Analytic Services builds the Size attribute dimension and associates its members with the appropriate members of the base dimension. Figure 140 shows the updated outline.

Figure 140: Multilevel Attribute Dimension

```
Database: Multilev
    Product {Size }
        100
            100-A23 {Size:7 }
            100-B54 {Size:8 }
        300
            300-R89 {Size:8 }
            300-U65 {Size:10 }
        400
            400-J43 {Size:9 }
    Size Attribute (Type: Numeric)
        Women
            7
            8
        Men
            8
            10
            9
```

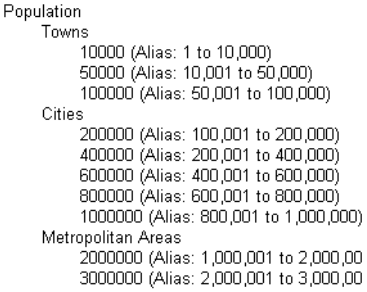
Working with Numeric Ranges

In many cases, you can use one rules file in a single dimension build operation to dynamically build attribute dimensions for numeric ranges and to associate the members of the base dimension with the ranges. However, in the following situations you must use two rules files, one to build the attribute dimension and one to associate the attributes with the appropriate members of the base dimension:

- When the range size is different for different members. For example, you can define small ranges for towns and cities with smaller populations, larger ranges for mid-sized cities, and ranges above 1,000,000 for cities with large populations.
- When the ranges are members of a multilevel attribute dimension. For example, the Population attribute dimension can have level 1 members that categorize the population ranges as Towns, Cities, and Metropolitan Areas.

The Population attribute dimension shown in Figure 141 demonstrates both situations. Population is a multilevel, numeric attribute dimension with level 0 members representing ranges of different sizes.

Figure 141: Numeric Attribute Dimension with Different-Sized Ranges



You must use one rules file to build the Population dimension and another rules file to associate the Population dimension members as attributes of members of the base dimension.

Building Attribute Dimensions that Accommodate Ranges

First, create a rules file that uses the generation, level, or parent-child build method to build the attribute dimension. In the rules file, be sure to specify the following:

- The name of the attribute dimension and its associated base dimension.
- The fields for building the attribute dimension. For a brief discussion and references to pertinent topics, see "Setting Field Type Information" on page 381.

The source data must be in attribute sequence, in ascending order. If ranges have different sizes, the source data must include a record for every attribute range.

Note: In later builds you cannot insert attribute members between existing members.

To use the generation method to build the outline in Figure 141, you must sequence the source data in ascending sequence, based on the numeric attribute value. Define the fields in a rules file as shown in Figure 142.

Figure 142: Rules File for Building a Numeric Attribute Dimension with Ranges

```
1    Towns■10000■<=10,000
2    Towns■50000■10,001 to 50,000
3    Towns■100000■50,001 to 100,000
4    Cities■200000■100,001 to 200,000
5    Cities■400000■200,001 to 400,000
6    Cities■500000■400,001 to 600,000
7    Cities■800000■600,001 to 800,000
8    Cities■1000000■800,001 to 1,000,000
9    Metropolitan Areas■2000000■1,000,001 to 2,000,000
10   Metropolitan Areas■3000000■2,000,001 to 3,000,000
```

	GEN2,Population	GEN3,Population	ALIAS3,Population
1	Towns	10000	<=10,000
2	Towns	50000	10,001 to 50,000
3	Towns	100000	50,001 to 100,000
4	Cities	200000	100,001 to 200,000
5	Cities	400000	200,001 to 400,000
6	Cities	500000	400,001 to 600,000
7	Cities	800000	600,001 to 800,000
8	Cities	1000000	800,001 to 1,000,000
9	Metropolitan Areas	2000000	1,000,001 to 2,000,000
10	Metropolitan Areas	3000000	2,000,001 to 3,000,000

Figure 142 also shows how you can associate aliases with attributes.

Associating Base Dimension Members with Their Range Attributes

After you build the numeric attribute dimension ranges, you need a rules file to associate the members of the base dimension with their attributes. The source data includes fields for the members of the base dimension and fields for the data values that Analytic Services uses to associate the appropriate Population attribute.

Define the rules file as shown in Figure 143.

Figure 143: Rules File for Associating Numeric Range Attributes

```
1    South■Albany, GA■117286
2    East■Boston, MA■3227707
3    East■Hartford, CT■1144574
4    West■Oakland, CA■2209629
5    Central■Rapid City, SD■87145
6    Central■St. Joseph, MO■97336
7    West■Tacoma, WA■657272
```

	GEN1,Market	GEN2,Market	Population3,Market
1	South	Albany, GA	117286
2	East	Boston, MA	3227707
3	East	Hartford, CT	1144574
4	West	Oakland, CA	2209629
5	Central	Rapid City, SD	87145
6	Central	St. Joseph, MO	97336
7	West	Tacoma, WA	657272

When you define the association field (for example, Population3, Market) be sure to place the attribute members within a range. In Data Prep Editor, in the Field Properties dialog box, on the Dimension Building Properties tab, click the Ranges button. Select “Place attribute members within a range.”

Note: Figure 143 includes a city, Boston, whose population of 3,227,707 is outside the ranges of the attribute dimension in Figure 141 on page 441. (The ranges in Figure 141 extend only to 3,000,000.)

To allow for values in the source data that are outside the ranges in the attribute dimension, enter a range size, such as 1000000. Analytic Services uses the range size to add members to the attribute dimension above the existing highest member or below the existing lowest member, as needed.

CAUTION: After you associate members of the base dimension with members of the attribute dimension, be aware that if you manually insert new members into the attribute dimension or rename members of the attribute dimension, you may invalidate existing attribute associations.

Consider an example where numeric range attributes are defined as "Tops of ranges" and an attribute dimension contains members 100, 200, 500, and 1000. A base dimension member with the value 556 is associated with the attribute 1000. If you rename a member of the attribute dimension from 500 to 600, the base dimension member with the value 556 now has an invalid association. This base member is still associated with the attribute 1000 when it should now be associated with the attribute 600.

If you manually insert new members or rename existing members, to ensure that associations are correct, rerun the dimension build procedure and associate the base members with the changed attribute dimensions. For example, rerunning the attribute association procedure correctly associates the member of the base dimension with the value 556 with the new attribute 600.

Assuring the Validity of Associations

To ensure the validity of attribute associations, you must be careful to select the correct dimension building options and to perform the builds in the proper sequence.

Adding or Changing Members of the Attribute Dimension: After you associate members of a base dimension with their numeric attribute ranges, if you manually insert new members or rename existing members in the attribute dimension, you should make sure that associations between attributes and base members are correct. To ensure that the associations are correct, you can do one of the following:

- Rerun the dimension build procedure that associates the base members with the changed attribute dimension.
- Use Outline Editor to manually review and fix, as needed, the associations of all base dimensions.

Deleting Members from the Attribute Dimension: You can delete all members of an attribute dimension so you can rebuild the dimension with new data. In Data Prep Editor, on the Dimension Building Properties tab in the Field Properties

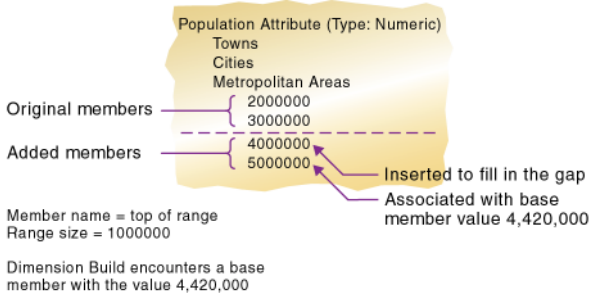
dialog box, click the Ranges button. Select “Delete all members of this attribute dimension.” Analytic Services uses the start value and range size value to rebuild the attribute dimension. To ensure proper attribute association, on the Dimension Build Settings tab of the Dimension Build Settings dialog box, for the base dimension you must select the “Allow Association Chgs” option.

Adding Members to the Base Dimension: You can use the same rules file to add new members to the base dimension and to associate the new members with their numeric range attributes simultaneously. Be sure to provide a value for the range size. In Data Prep Editor, on the Dimension Building Properties tab in the Field Properties dialog box, click the Ranges button and specify the range size for the attribute dimension.

If Analytic Services encounters a base dimension value that is greater than the highest attribute member by more than the range size or is lower than the lowest attribute member by more than the range size, it creates members in the attribute dimension to accommodate the out-of-range values.

Consider the example, in Figure 141 on page 441, where numeric range attributes are defined as “Tops of ranges.” The highest value member of the Population attribute dimension is 3000000. If the source data includes a record with the population 4,420,000 and the range size is 1000000, Analytic Services adds two members to the attribute dimension, 4000000 and 5000000, and associates the base member with the 5000000 attribute.

Figure 144: Dynamically Adding Attribute Range Members



When you add range members and base dimension members at the same time, Analytic Services does not create aliases for the new members of the attribute dimension. If you want aliases that describe the range values for the new members of the attribute dimension, you must add the aliases in a separate operation.

Reviewing the Rules for Building Attribute and Base Dimensions

The following list describes a few areas unique to defining and associating attributes through dimension build.

Getting Ready

- Before running a dimension build, you must define the attribute member name formats for the outline. For a comprehensive discussion of assigning member names in attribute dimensions, see "Setting Member Names in Attribute Dimensions" on page 196.
- Defining new attribute dimensions in a rules file is different from defining new standard dimensions in a rules file.

Defining Fields in Rules Files

Rules files that are used to build single-level attribute dimensions require fewer field types than rules files that build and associate members of multilevel attribute dimensions.

- For single-level attribute dimensions, define the field that contains the attribute values as the field to be associated with the members of the base dimension. A dimension build uses the defined field to add new members to the attribute dimension. For a description of how to define fields, see "Associating Attributes" on page 436.
- For multilevel attribute dimensions, Analytic Services requires fields that define each generation or level in the attribute dimension and fields that define the associations. Use the new field type, Attribute Parent, to identify fields that are parent members for the attribute members being associated. For a description of how to handle multilevel attribute dimensions, see "Working with Multilevel Attribute Dimensions" on page 438.

Controlling Adding New Attribute Members

When Analytic Services encounters attribute data values that are not members of the attribute dimension, it automatically adds the values as new members. To prevent adding new members to attribute dimensions, do either of the following:

In the Dimension Build Settings dialog box, select the Do Not Create Mbrs option for the attribute dimension. For more information, see "Setting Member Properties" in the *Essbase Administration Services Online Help*.

Controlling Associations

Association to Control	How to Control the Association
Making changes to attribute associations	In Data Prep Editor, on the Dimension Build Settings tab of the Dimension Build Settings dialog box, select the Allow Association Chgs option for the attribute dimension. For more information, see "Setting Member Properties" in the *Essbase Administration Services Online Help*.
Enabling automatic association of base members with attributes that represent ranges of values	In Data Prep Editor, on the Dimension Building Properties tab in the Field Properties dialog box, click the Ranges button and define the size of the range. For a brief discussion of field types and references to pertinent topics, see "Setting Field Type Information" on page 381.
Concurrent attribute associations	Use any build method except the Add as build methods. For information about each build method, see Table 22 on page 420.

Note: Because attributes are defined only in the outline, the data load process does not affect them.

Building Shared Members by Using a Rules File

The data associated with a shared member comes from a real member with the same name as the shared member. The shared member stores a pointer to data contained in the real member; thus the data is shared between the members and is stored only one time.

In the Sample Basic database, for example, the 100-20 (Diet Cola) member rolls up into the 100 (Cola) family and into the Diet family.

Figure 145: Shared Members in the Sample Basic Database

You can share members among as many parents as you want. Diet Cola has two parents, but you can define it to roll up into even more parents.

You can share members at multiple generations in the outline. In Figure 145, Diet Cola is shared by two members at generation 2 in the outline, but it can be shared by a member at generation 3 and a member at generation 4 as in Figure 153.

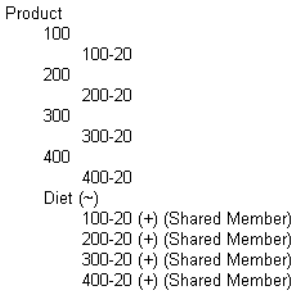
Creating shared members at different generations in the outline is easy in Outline Editor. However, creating shared members using dimension build is a little more difficult. You must pick the build method and format the data source carefully. The following sections describe how to build shared members in the outline by using a data source and a rules file.

Note: You should not create an outline in which a shared member is located before the actual member with which it is associated.

Sharing Members at the Same Generation

Members that are shared at the same generation roll up into the same branch. In the Sample Basic database, 100-20 (Diet Cola) is shared by two parents. Both parents roll up into the same branch, that is, the Product dimension, and both parents are at generation 2.

Figure 146: Members Shared at the Same Generation



This scenario is the simplest way to share members. You can share members at the same generation by using any of these build methods. These methods are discussed in the following sections:

- "Using Generation References to Create Same Generation Shared Members" on page 449
- "Using Level References to Create Same Generation Shared Members" on page 450
- "Using Parent-Child References to Create Same Generation Shared Members" on page 451

Using Generation References to Create Same Generation Shared Members

To create shared member parents at the same generation by using the generation references build method, define the field type for the parent of the shared members as DUPGEN. A *duplicate generation* is a generation with shared members for children. Use the same GEN number as the primary member.

For example, to create the Diet parent and share the 100-20, 200-20, 300-20, and 400-20 members, use the sample file, SHGENREF.TXT, and set up the rules file so that the fields look like SHGENREF.RUL, shown in Figure 147. Remember 100 is the Cola family, 200 is the Root Beer family, 300 is the Cream Soda family, and the -20 after the family name indicates a diet version of the soda.

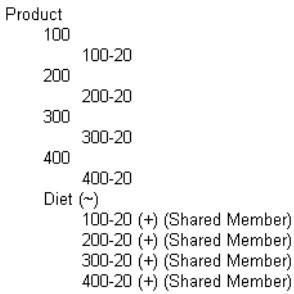
Figure 147: Sample Generation Shared Member Rules File

```
1    100■Diet■100-20
2    200■Diet■200-20
3    300■Diet■300-20
4    400■Diet■400-20
```

	GEN2,Product	DUPGEN2,Product	GEN3,Product
1	100	Diet	100-20
2	200	Diet	200-20
3	300	Diet	300-20
4	400	Diet	400-20

The data source and rules file illustrated in Figure 147 build the following tree:

Figure 148: Sample Generation Shared Member Rules Tree



Using Level References to Create Same Generation Shared Members

To create shared members of the same generation by using the level references build method, first make sure that the primary and any secondary roll-ups are specified in one record. You can specify as many secondary roll-ups as you want, as long as the roll-ups are all in one record.

Define the field type for the shared member as LEVEL. Then enter the level number. To create a shared member of the same generation, set the level number of the secondary roll-up to have the same number of levels as the primary roll-up. While processing the data source, Analytic Services creates a parent at the specified level and inserts the shared members under it.

For example, to create the shared 100-20 (Diet Cola), 200-20 (Diet Root Beer), 300-20 (Diet Cream Soda), and 400-20 (Fruit Soda) members in the Sample Basic database, use the sample file, SHLEV.TXT, and set up the rules file so that the fields look like SHLEV.RUL shown in Figure 149.

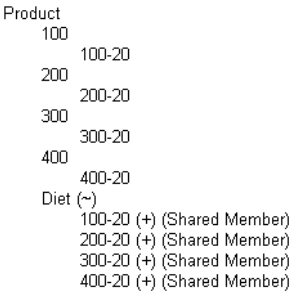
Figure 149: Sample Level Shared Member Rules File

```
1    100-20■100■Diet
2    200-20■200■Diet
3    300-20■300■Diet
4    400-20■400■Diet
```

	LEVEL0,Product	LEVEL1,Product	LEVEL1,Product
1	100-20	100	Diet
2	200-20	200	Diet
3	300-20	300	Diet
4	400-20	400	Diet

The data source and rules file illustrated in Figure 149 build the following tree:

Figure 150: Sample Level Shared Member Rules Tree

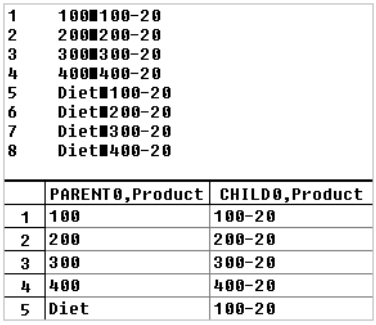


Using Parent-Child References to Create Same Generation Shared Members

To create shared members of the same generation by using the parent-child references build method, define the PARENT and CHILD field types. Make sure that Analytic Services is set up to allow sharing (clear Do Not Share in the

Dimension Build Settings tab of the Dimension Build Settings dialog box). When sharing is enabled, Analytic Services automatically creates duplicate members under a new parent as shared members.

Figure 151: Sample Parent-Child Shared Members Rules File

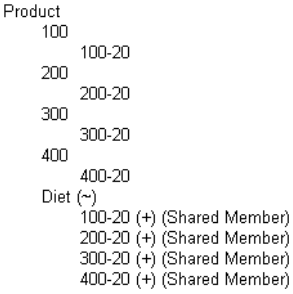


```
1    100■100-20
2    200■200-20
3    300■300-20
4    400■400-20
5    Diet■100-20
6    Diet■200-20
7    Diet■300-20
8    Diet■400-20
```

	PARENT0,Product	CHILD0,Product
1	100	100-20
2	200	200-20
3	300	300-20
4	400	400-20
5	Diet	100-20

The data source and rules file illustrated in Figure 151 build the following tree:

Figure 152: Sample Parent-Child Shared Member Rules Tree

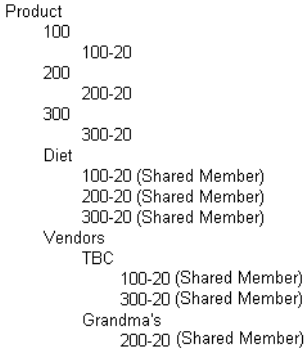


```
Product
    100
        100-20
    200
        200-20
    300
        300-20
    400
        400-20
    Diet (~)
        100-20 (+) (Shared Member)
        200-20 (+) (Shared Member)
        300-20 (+) (Shared Member)
        400-20 (+) (Shared Member)
```

Sharing Members at Different Generations

Sometimes you want shared members to roll up into parents that are at different generations in the outline. In Figure 153, for example, the shared members roll up into parents at generation 2 and at generation 3. This outline assumes that The Beverage Company (TBC) buys some of its beverages from outside vendors. In this case, it buys 200-20 (Diet Root Beer) from a vendor named Grandma's.

Figure 153: Members Shared at Different Generations



To share members across parents at different generations in the outline, use one of these build methods. The methods are described in the following sections:

- "Using Level References to Create Different Generation Shared Members" on page 453
- "Using Parent-Child References to Create Different Generation Shared Members" on page 454

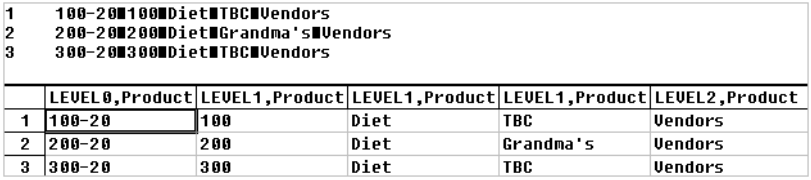
Using Level References to Create Different Generation Shared Members

To create shared members of different generations by using the level references build method, first make sure that both primary and secondary roll-ups are specified in one record. You can specify as many secondary roll-ups as you want, as long as the roll-ups are all in one record.

Define the field type for the shared member as LEVEL. Then enter the level number. While processing the data source, Analytic Services creates a parent at the specified level and inserts the shared members under it.

For example, to share the products 100-20, 200-20, and 300-20 with a parent called Diet and two parents called TBC (The Beverage Company) and Grandma's, use the sample data file and the rules file in Figure 154.

Figure 154: Level References Sample Rules File for Shared Members at Different Generations



```
1    100-20■100■Diet■TBC■Vendors
2    200-20■200■Diet■Grandma's■Vendors
3    300-20■300■Diet■TBC■Vendors
```

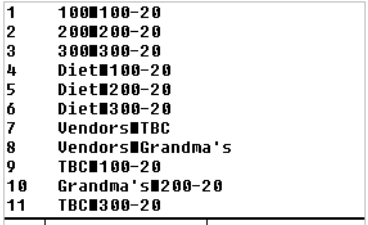
	LEVEL0,Product	LEVEL1,Product	LEVEL1,Product	LEVEL1,Product	LEVEL2,Product
1	100-20	100	Diet	TBC	Vendors
2	200-20	200	Diet	Grandma's	Vendors
3	300-20	300	Diet	TBC	Vendors

The data source and rules file illustrated in Figure 154 build the tree illustrated in Figure 153.

Using Parent-Child References to Create Different Generation Shared Members

To create shared members at the different generation using the parent-child references build method, define the PARENT and CHILD field types. Make sure that Analytic Services is set up to allow sharing (clear Do Not Share in the Dimension Build Settings tab of the Dimension Build Settings dialog box). When sharing is enabled, Analytic Services automatically creates duplicate members under a new parent as shared members.

Figure 155: Parent-Child References Sample Rules File for Shared Members at Different Generations



```
1    100■100-20
2    200■200-20
3    300■300-20
4    Diet■100-20
5    Diet■200-20
6    Diet■300-20
7    Vendors■TBC
8    Vendors■Grandma's
9    TBC■100-20
10   Grandma's■200-20
11   TBC■300-20
```

	PARENT0,Produc	CHILD0,Prod
1	100	100-20
2	200	200-20
3	300	300-20
4	Diet	100-20
5	Diet	200-20
6	Diet	300-20

The data source and rules file illustrated in Figure 155 build the tree illustrated in Figure 153.

Sharing Non-Leaf Members

Sometimes you want to share non-leaf members (members that are not at the lowest generation). In Figure 156 for example, 100, 200, and 300 are shared by TBC and Grandma's. This outline assumes that TBC (The Beverage Company) buys some of its product lines from outside vendors. In this case, it buys 200 (all root beer) from a vendor named Grandma's.

Figure 156: Non-Leaf Members Shared at Different Generations

```
Product
    Soda
        100
            100-20
        200
            200-20
        300
            300-20
        Diet
            100-20 (Shared Member)
            200-20 (Shared Member)
            300-20 (Shared Member)
    Vendors
        TBC
            100 (Shared Member)
            300 (Shared Member)
        Grandma's
            200 (Shared Member)
```

To share non-leaf members, use one of these build methods. These methods are described in the following sections:

- "Using Level References to Create Non-Leaf Shared Members" on page 455
- "Using Parent-Child References to Create Non-Leaf Shared Members" on page 456

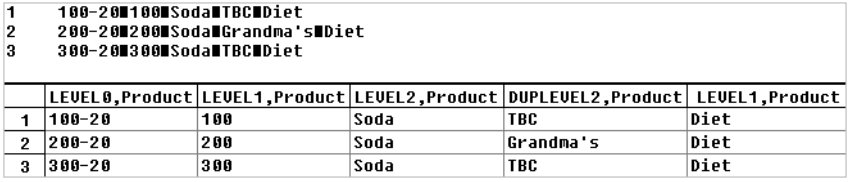
Using Level References to Create Non-Leaf Shared Members

To create shared non-leaf members by using the level references build method, first make sure that both primary and secondary roll-ups are specified in one record. You can specify as many secondary roll-ups as you want, as long as the roll-ups are all in one record.

Define the field type for the parent of the shared member as duplicate level (DUPLEVEL). Then enter the level number. To create a shared member of the same generation, set the level number of the secondary roll-up to have the same number of levels as the primary roll-up. While processing the data source, Analytic Services creates a parent at the specified level and inserts the shared members under it.

For example, to share the product lines 100, 200, and 300 with a parent called Soda and two parents called TBC and Grandma's, use the sample data file and rules file shown in Figure 157. This data source and rules file work only if the Diet, TBC, and Grandma's members exist in the outline. The DUPLEVEL field is always created as a child of the dimension (that is, at generation 2), unless the named level field already exists in the outline.

Figure 157: Level References Sample Rules File for Non-Leaf Shared Members at Different Generations



```
1    100-20■100■Soda■TBC■Diet
2    200-20■200■Soda■Grandma's■Diet
3    300-20■300■Soda■TBC■Diet
```

	LEVEL0,Product	LEVEL1,Product	LEVEL2,Product	DUPLEVEL2,Product	LEVEL1,Product
1	100-20	100	Soda	TBC	Diet
2	200-20	200	Soda	Grandma's	Diet
3	300-20	300	Soda	TBC	Diet

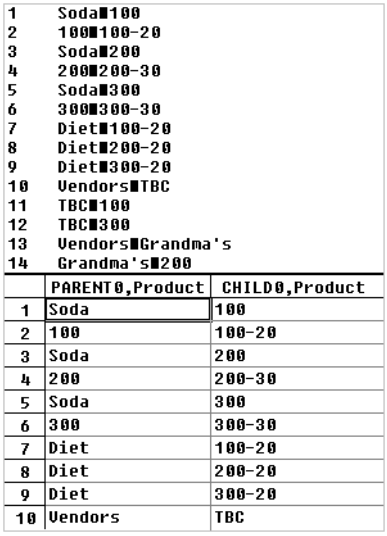
The data source and rules file illustrated in Figure 157 build the tree illustrated in Figure 156.

Using Parent-Child References to Create Non-Leaf Shared Members

To create shared non-leaf members at the same generation using the parent-child references build method, define the PARENT and CHILD field types. Make sure that Analytic Services is set up to allow sharing (clear Do Not Share in the Dimension Build Settings tab of the Dimension Build Settings dialog box). When sharing is enabled, Analytic Services automatically creates duplicate members under a new parent as shared members.

The parent-child references build method is the most versatile for creating shared members. It does not have any restrictions on the position of the shared members in the outline, unlike the generation references and level references build methods.

Figure 158: Parent-Child Sample Rules File for Non-Leaf Shared Members



```
1    Soda■100
2    100■100-20
3    Soda■200
4    200■200-30
5    Soda■300
6    300■300-30
7    Diet■100-20
8    Diet■200-20
9    Diet■300-20
10   Vendors■TBC
11   TBC■100
12   TBC■300
13   Vendors■Grandma's
14   Grandma's■200
```

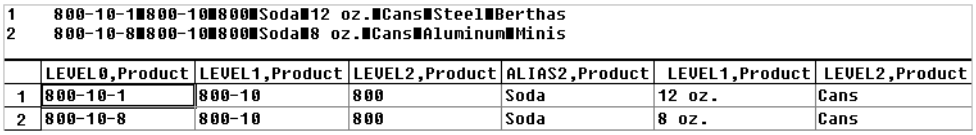
	PARENT0,Product	CHILD0,Product
1	Soda	100
2	100	100-20
3	Soda	200
4	200	200-30
5	Soda	300
6	300	300-30
7	Diet	100-20
8	Diet	200-20
9	Diet	300-20
10	Vendors	TBC

The data source and rules file illustrated in Figure 158 build the tree illustrated in Figure 156.

Building Multiple Roll-Ups by Using Level References

To enable the retrieval of totals from multiple perspectives, you can also put shared members at different levels in the outline. Use the level references build method. The rules file, `LEVELMUL.RUL`, in Figure 159 specifies an example of build instructions for levels in the Product dimension.

Figure 159: Rules File Fields Set to Build Multiple Roll-Ups Using Level References



```
1    800-10-1■800-10■800■Soda■12 oz.■Cans■Steel■Berthas
2    800-10-8■800-10■800■Soda■8 oz.■Cans■Aluminum■Minis
```

	LEVEL0,Product	LEVEL1,Product	LEVEL2,Product	ALIAS2,Product	LEVEL1,Product	LEVEL2,Product
1	800-10-1	800-10	800	Soda	12 oz.	Cans
2	800-10-8	800-10	800	Soda	8 oz.	Cans

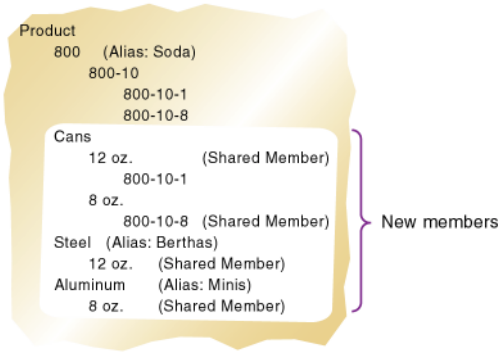
Because the record is so long, this second graphic shows the rules file after it has been scrolled to the right to show the extra members:

Figure 160: Scrolled Window

DUPLEVEL2,Product	DUPLEVALIAS2,Product
Steel	Berthas
Aluminum	Minis

When you run the dimension build using the data in Figure 159, Analytic Services builds the following member tree:

Figure 161: Multiple Roll-Ups



This example enables analysis not only by package type (Cans), but also by packaging material; for example, analysis comparing sales of aluminum cans and steel cans.

Because Product is a sparse dimension, you can use an alternative outline design to enable retrieval of the same information. Consider creating a multilevel attribute dimension for package type with Steel and Aluminum as level 0 members under Can. For a discussion of outline design guidelines, see “Analyzing Database Design” on page 88.

Creating Shared Roll-Ups from Multiple Data Sources

In many situations, the data for a dimension is in two or more data sources. If you are building dimensions from more than one data source and want to create multiple roll-ups, load the first data source using the most appropriate build method and then load all other data sources using the parent-child references build

method. Make sure that Analytic Services is set up to allow sharing (clear Do Not Share in the Dimension Build Settings tab of the Dimension Build Settings dialog box).

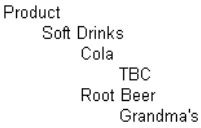
For example, using the Product data source in Figure 162:

Figure 162: Soft Drinks Data Source

```
"Soft Drinks"   Cola
"Soft Drinks"   "Root Beer"
Cola            TBC
"Root Beer"     Grandma's
```

Analytic Services builds the tree illustrated in Figure 163:

Figure 163: Soft Drinks Tree



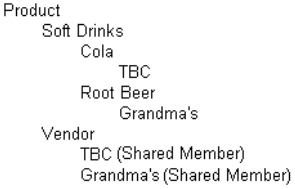
Then load the second data source, illustrated in Figure 164, to relate the products to the vendors using the parent-child build method. Make sure that Analytic Services is set up to allow sharing.

Figure 164: Second Shared Roll-Ups Data Source

```
Vendor    TBC
Vendor    Grandma's
```

Analytic Services builds the tree illustrated in Figure 165:

Figure 165: Shared Roll-Ups Tree



Part IV

Calculating Data

This part describes how to calculate the data in Analytic Services databases, including how to create formulas, define calculation order, calculate data values dynamically, calculate time series data, and create calculation scripts:

- Chapter 21, "Calculating Analytic Services Databases," explains the basic concepts behind database calculations.
- Chapter 22, "Developing Formulas," explains formulas and describes how to use Formula Editor to create formulas on members.
- Chapter 23, "Reviewing Examples of Formulas," contains detailed examples of formulas.
- Chapter 24, "Defining Calculation Order," describes how to set the calculation order of the members in a database.
- Chapter 25, "Dynamically Calculating Data Values," describes how to set Essbase to calculate values for dimensions and members when they are requested by users rather than as a part of database consolidations.
- Chapter 26, "Calculating Time Series Data," describes how to calculate time series data, including First, Last, Average, and Period-To-Date values, for both single server and partitioned applications.
- Chapter 27, "Developing Calculation Scripts," explains calculation scripts and describes how to use Calculation Script Editor to create calculation scripts.
- Chapter 28, "Reviewing Examples of Calculation Scripts," contains detailed examples of calculation scripts.

- Chapter 29, “Developing Custom-Defined Calculation Macros,” explains how to combine multiple calculation functions into single macro functions to be used in calculation scripts and formulas.
- Chapter 30, “Developing Custom-Defined Calculation Functions,” explains how to develop custom calculation functions not otherwise supported by the Analytic Services calculation scripting language.

Note: For optimization information, see Chapter 54, “Optimizing Calculations” and Chapter 55, “Optimizing with Intelligent Calculation.”

Chapter 21

Calculating Analytic Services Databases

This chapter explains the basic concept of multidimensional database calculation and provides information about how to calculate an Analytic Services block storage database.

This chapter includes the following sections:

Note: Most computers represent numbers in binary, and therefore can only represent real numbers approximately. Because binary computers cannot hold an infinite number of bits after a decimal point, numeric fractions such as one third (0.3333...), cannot be expressed as a decimal with a terminating point. Fractions with a denominator of the power of two (for example, 0.50) or ten (0.10) are the only real numbers that can be represented exactly. For details, see IEEE Standard 754 for Floating-Point Representation (IEEE, 1985).

For information about calculating aggregate storage databases see "Calculating Aggregate Storage Databases" on page 1333.

About Database Calculation

A database contains two types of values. It contains the values that you enter, which are called *input data*, and the values that are calculated from the input data.

Consider the following examples:

- You enter regional sales figures for a variety of products. You calculate the total sales for each product.
- You enter the budget and actual values for the cost of goods sold for several products in several regions. You calculate the variance between budget and actual values for each product in each region.
- The database contains regional sales figures and prices for all products. You calculate what happens to total profit if you increase the price of one product in one region by 5%.

Small differences in the precision of cell values may occur between calculations run on different platforms, due to operating system math library differences.

Analytic Services offers two ways that you can calculate a database:

- Outline calculation
- Calculation script calculation

Which way you choose depends on the type of calculation that you want to do.

Outline Calculation

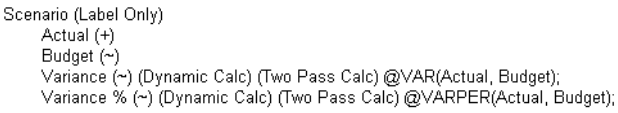
Outline calculation is the simplest method of calculation. Analytic Services bases the calculation of the database on the relationships between members in the database outline and on any formulas that are associated with members in the outline.

For example, Figure 166 shows the relationships between the members of the Market dimension in the Sample Basic database. The values for New York, Massachusetts, Florida, Connecticut, and New Hampshire are added to calculate the value for East. The values for East, West, South, and Central are added to calculate the total value for Market.

Figure 166: Relationship Between Members of the Market Dimension

Figure 167 shows the Scenario dimension from the Sample Basic database. The Variance and Variance % members are calculated by using the formulas attached to them.

Figure 167: Calculation of Variance and Variance %



It may be more efficient to calculate some member combinations when you retrieve the data, instead of calculating the member combinations during the regular database calculation. You can use dynamic calculations to calculate data at retrieval time. For a comprehensive discussion of dynamic calculation, see Chapter 25, “Dynamically Calculating Data Values.”

Calculation Script Calculation

Calculation script calculation is the second method of calculation. Using a calculation script, you can choose exactly how to calculate a database. For example, you can calculate part of a database or copy data values between members.

A calculation script contains a series of calculation commands, equations, and formulas. For example, the following calculation script increases the actual marketing expenses in the New York region by 5%.

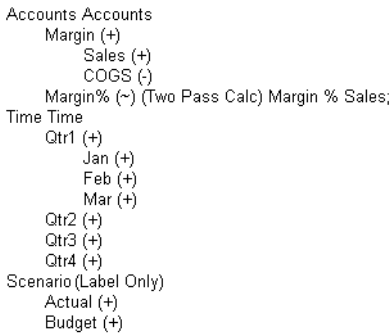
```
FIX (Actual, “New York”)
     Marketing = Marketing *1.05;
ENDFIX;
```

For a comprehensive discussion of calculation scripts, see Chapter 27, “Developing Calculation Scripts.”

About Multidimensional Calculation Concepts

For an illustration of the nature of multidimensional calculations, consider the following, simplified database:

Figure 168: Calculating a Multidimensional Database



The database has three dimensions—Accounts, Time, and Scenario.

The Accounts dimension has four members:

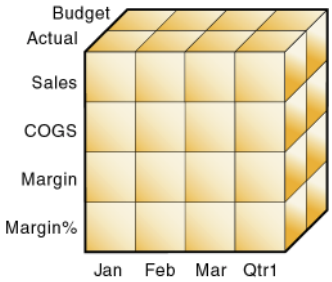
- Sales and COGS are input values.
- Margin = Sales - COGS.
- Margin% = Margin % Sales (Margin as a percentage of Sales).

The Time dimension has four quarters. The example displays only the members in Qtr1—Jan, Feb, and Mar.

The Scenario dimension has two child members—Budget for budget values and Actual for actual values.

An intersection of members (one member on each dimension) represents a data value. Our example has three dimensions; therefore, the dimensions and data values in the database can be represented as a cube, as shown in Figure 169:

Figure 169: Three-Dimensional Database

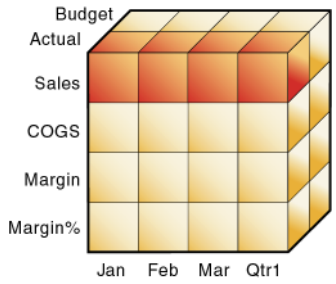


As shown in Figure 170, when you refer to Sales, you are referring to a slice of the database containing eight Sales values.

Figure 170: Sales, Actual, Budget Slice of the Database

As shown in Figure 171, when you refer to Actual Sales, you are referring to four Sales values:

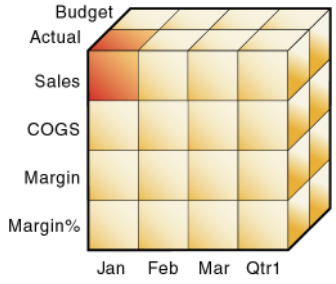
Figure 171: Actual, Sales Slice of the Database



To refer to a specific data value in a multidimensional database, you need to specify each member on each dimension. A data value is stored in a single cell in the database. In Figure 172, the cell containing the data value for Sales, Jan, Actual is shaded.

In Analytic Services, member combinations are denoted by a cross-dimensional operator. The symbol for the cross-dimensional operator is ->. So Sales, Jan, Actual is written Sales -> Jan -> Actual.

Figure 172: Sales, Jan, Actual Slice of the Database



When Analytic Services calculates the formula “Margin% = Margin % Sales,” it takes each Margin value and calculates it as a percentage of its corresponding Sales value.

Analytic Services cycles through the database and calculates Margin% as follows:

1. Margin -> Jan -> Actual as a percentage of Sales -> Jan -> Actual. The result is placed in Margin% -> Jan -> Actual.
2. Margin -> Feb -> Actual as a percentage of Sales -> Feb -> Actual. The result is placed in Margin% -> Feb -> Actual.
3. Margin -> Mar -> Actual as a percentage of Sales -> Mar -> Actual. The result is placed in Margin% -> Mar -> Actual.
4. Margin -> Qtr1 -> Actual as a percentage of Sales -> Qtr1 -> Actual. The result is placed in Margin% -> Qtr1 -> Actual.
5. Margin -> Jan -> Budget as a percentage of Sales -> Jan -> Budget. The result is placed in Margin% -> Jan -> Budget.
6. Analytic Services continues cycling through the database until it has calculated Margin% for every combination of members in the database.

For a comprehensive discussion of how Analytic Services calculates a database, see Chapter 24, “Defining Calculation Order.”

Setting the Default Calculation

By default, the calculation for a database is a CALC ALL of the database outline. CALC ALL consolidates all dimensions and members and calculates all formulas in the outline.

However, you can specify any calculation script as the default database calculation. Thus, you can assign a frequently-used script to the database rather than loading the script each time you want to perform its calculation. Also, if you want a calculation script to work with calculation settings defined at the database level, you must set the calculation script as the default calculation.

➤ Use any of the following methods to set the default calculation:

Tool	Topic	Location
Administration Services	Setting the Default Calculation	*Essbase Administration Services Online Help*
MaxL	**alter database**	*Technical Reference*
ESSCMD	SETDEFAULTCALCFILE	*Technical Reference*

Calculating Databases

If you have Calculation permissions, you can calculate a database. When you use Essbase Administration Services to calculate a database, you can execute the calculation in the background so that you can continue working as the calculation processes. You can then check the status of the background process to see when the calculation is complete. For instructions, see "Calculating Block Storage Databases" in *Essbase Administration Services Online Help*.

➤ Use any of the following methods to calculate a database:

Tool	Topic	Location
Administration Services	Calculating Block Storage Databases	*Essbase Administration Services Online Help*
MaxL	**execute calculation**	*Technical Reference*
ESSCMD	CALC, CALCDEFAULT, and CALCLINE	*Technical Reference*
Spreadsheet Add-in	Calculating a Database	*Spreadsheet Add-in Online Help*

Canceling Calculations

➤ To stop a calculation before Analytic Services completes it, click the Cancel button while the calculation is running.

When you cancel a calculation, Analytic Services performs one of the following operations:

- Reverts all values to their previous state
- Retains any values calculated before the cancellation

How Analytic Services handles the cancellation depends on the Analytic Services Kernel Isolation Level settings. For a description of these settings, see "Understanding Isolation Levels" on page 1054.

Parallel and Serial Calculation

Analytic Services now supports parallel calculation in addition to serial calculation. Serial calculation, the default, means that all steps in a calculation run on a single thread. Each task is completed before the next is started. Parallel calculation means that the Analytic Services calculator can analyze a calculation, and, if appropriate, assign tasks to multiple CPUs (up to 4).

For a comprehensive discussion of parallel calculation, including how to determine whether Analytic Server should use parallel calculation, see "Using Parallel Calculation" on page 1182.

Security Considerations

In order to calculate a database, you must have Calculate permissions for the database outline. If you have calculate permissions, you can calculate any value in the database. With calculate permissions, you can calculate a value even if a security filter denies you read and update permissions. Careful consideration should be given to providing users with calculate permissions.

For information on providing users with calculate permissions and on security filters, see Chapter 36, "Managing Security for Users and Applications."

Chapter 22

Developing Formulas

This chapter explains how to develop and use formulas to calculate a database. It provides detailed examples of formulas, which you may want to adapt for your own use. For more examples, see Chapter 23, "Reviewing Examples of Formulas."

The information in this chapter does not apply to aggregate storage outlines. For information about developing formulas in MDX for aggregate storage outline members, see "Developing Formulas on Aggregate Storage Outlines" on page 1309.

This chapter includes the following topics:

Using formulas can have significant implications for calculation performance. After reading this chapter, use the information in Chapter 54, "Optimizing Calculations" to design and create formulas optimized for performance.

For information on using formulas with Hybrid Analysis, see "Using Formulas with Hybrid Analysis" on page 304.

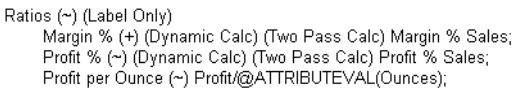
Understanding Formulas

Formulas calculate relationships between members in a database outline. You can use formulas in two ways:

- Apply them to members in the database outline. Use this method if you do not need to control database calculations carefully for accuracy or performance. This method limits formula size to less than 64 kilobytes. For instructions, see "Composing Formulas" on page 483.
- Place them in a calculation script. Use this method if you need to control database calculations carefully. For more information, see "Using Formulas in a Calculation Script" on page 589.

The following figure shows the Measures dimension from the Sample Basic database. The Margin %, Profit %, and Profit per Ounce members are calculated using the formulas applied to them.

Figure 173: Calculation of Margin %, Profit %, and Profit per Ounce



Analytic Services provides a comprehensive set of operators and functions, which you can use to construct formula calculations on a database. The rest of this section provides a description of the elements you can place in a formula, and provides basic information about formula calculation and syntax:

- "Operators" on page 475
- "Functions" on page 475
- "Dimension and Member Names" on page 478
- "Constant Values" on page 478
- "Non-Constant Values" on page 478

Operators

The following table shows the types of operators you can use in formulas:

Table 23: Descriptions of Operator Types

Operator Type	Description
Mathematical	Perform common arithmetic operations. For example, you can add, subtract, multiply, or divide values. For a complete list of the mathematical operators, see the *Technical Reference*.
Conditional	Control the flow of formula executions based on the results of conditional tests. For example, you can use an IF statement to test for a specified condition. For a list of the conditional operators, see the *Technical Reference*. For information on writing conditional formulas, see "Conditional Tests" on page 484.
Cross-dimensional	Point to the data values of specific member combinations. For example, point to the sales value for a specific product in a specific region. For examples of how to use the cross-dimensional operator, see "Working with Member Combinations across Dimensions" on page 499.

For information about using operators with #MISSING, zero, and other values, see the "Analytic Services Functions" section in the *Technical Reference*.

Functions

Functions are predefined routines that perform specialized calculations and return sets of members or data values. The following table shows the types of functions you can use in formulas.

For detailed examples of formulas, see Chapter 23, "Reviewing Examples of Formulas."

Table 24: Descriptions of Function Types

Function Type	Description
Boolean	Provide a conditional test by returning either a TRUE (1) or FALSE (0) value. For example, you can use the @ISMBR function to determine whether the current member is one that you specify.
Mathematical	Perform specialized mathematical calculations. For example, you can use the @AVG function to return the average value of a list of members.
Relationship	Look up data values within a database during a calculation. For example, you can use the @ANCESTVAL function to return the ancestor values of a specified member combination.
Range	Declare a range of members as an argument to another function or command. For example, you can use the @SUMRANGE function to return the sum of all members that lie within a specified range.
Financial	Perform specialized financial calculations. For example, you can use the @INTEREST function to calculate simple interest or the @PTD function to calculate period-to-date values.
Member Set	Generate a list of members that is based on a specified member. For example, you can use the @ICHILDREN function to return a specified member and its children.
Allocation	Allocate values that are input at a parent level across child members. You can allocate values within the same dimension or across multiple dimensions. For example, you can use the @ALLOCATE function to allocate sales values that are input at a parent level to the children of the parent; the allocation of each child is determined by its share of the sales of the previous year.

Table 24: Descriptions of Function Types (Continued)

Function Type	Description
Forecasting	Manipulate data for the purposes of smoothing or interpolating data, or calculating future values. For example, you can use the @TREND function to calculate future values that are based on curve-fitting to historical values.
Statistical	Calculate advanced statistics. For example, you can use the @RANK function to calculate the rank of a specified member or a specified value in a data set.
Date and Time	Use date and time characteristics in calculation formulas. For example, you can use the @TODATE function to convert date strings to numbers that can be used in calculation formulas.
Miscellaneous	This type provides two different kinds of functionality: • You can specify calculation modes that Analytic Services is to use to calculate a formula—cell, block, bottom-up, and top-down • You can manipulate character strings for member and dimension names; for example, to generate member names by adding a character prefix to a name or removing a suffix from a name, or by passing the name as a string.
Custom-Defined Functions	This type enables you to perform functions that you develop for calculation operations. These custom-developed functions are written in the Java programming language and are called by the Analytic Services calculator framework as external functions.

For a complete list of operators, functions, and syntax, see the *Technical Reference*.

Note: Abbreviations of functions are not supported. Some commands may work in an abbreviated form, but if there is another function with a similar name, Analytic Services may use the wrong function. Use the complete function name to ensure correct results.

Dimension and Member Names

You can include dimension and member names in a formula, as illustrated in the following example:

```
Scenario
100-10
Feb
```

Constant Values

You can assign a constant value to a member:

```
California = 120;
```

In this formula, California is a member in a sparse dimension and 120 is a constant value. Analytic Services automatically creates all possible data blocks for California and assigns the value 120 to all data cells. Many thousands of data blocks may be created. To assign constants in a sparse dimension to only those intersections that require a value, use FIX as described in “Constant Values Assigned to Members in a Sparse Dimension” on page 1196.

Non-Constant Values

If you assign anything other than a constant to a member in a sparse dimension, and no data block exists for that member, new blocks may not be created unless Analytic Services is enabled to create blocks on equations.

For example, to create blocks for West that didn’t exist prior to running the calculation, you need to enable Create Blocks on Equations for this formula:

```
West = California + 120;
```

You can enable Create Blocks on Equations at the database level whereby blocks are always created, or you can control block creation within calculation scripts.

➤ To enable the Create Blocks on Equations feature for all calculation scripts for a specific database, use any of the following methods:

Tool	Topic	Location
Administration Services	Enabling Create Blocks on Equations	*Essbase Administration Services Online Help*
MaxL	**alter database**	*Technical Reference*
ESSCMD	SETDBSTATE	*Technical Reference*

Because unnecessary blocks can be created when Create Blocks on Equations is enabled at the application or database level, calculation performance can be affected. To control block creation within a calculation script, use the SET CREATEBLOCKEQ ON|OFF calculation command as described in "Non-Constant Values Assigned to Members in a Sparse Dimension" on page 1197.

Understanding Formula Calculation

For formulas applied to members in a database outline, Analytic Services calculates formulas when you do the following:

- Run a default (CALC ALL) calculation of a database.
- Run a calculation script that calculates the member containing the formula; for example, a CALC DIM of the dimension containing the member, or the member itself. For information about how to develop calculation scripts and how to use them to control how Analytic Services calculates a database, see Chapter 27, "Developing Calculation Scripts."

For a formula in a calculation script, Analytic Services calculates the formula when it occurs in the calculation script.

If a formula is associated with a dynamically calculated member, Analytic Services calculates the formula when the user requests the data values. In a calculation script, you cannot calculate a dynamically calculated member or make a dynamically calculated member the target of a formula calculation. For an explanation of how you calculate data values dynamically and how you benefit from doing so, see Chapter 25, "Dynamically Calculating Data Values."

Using dynamically calculated members in a formula on a database outline or in a calculation script can significantly affect calculation performance. Performance is affected because Analytic Services has to interrupt the regular calculation to perform the dynamic calculation.

You cannot use substitution variables in formulas that you apply to the database outline. For an explanation of how substitution variables can be used, see “Using Substitution Variables” on page 494.

Understanding Formula Syntax

When you create member formulas, make sure the formulas follow these rules:

- End each statement in the formula with a semicolon (;). For example,

  ```
  Margin % Sales;
  ```

- Enclose a member name in double quotation marks ("") if the member name meets any of the following conditions:

  – Contains spaces; for example,

    "Opening Inventory" = "Ending Inventory" - Sales + Additions;

  – Is the same as an operator or function name. See the *Technical Reference* for a list of operators and functions.

  – Includes any non-alphanumeric character; for example, hyphens (-), asterisks (*), and slashes (/).

  – Is all numeric or starts with one or more numerals; for example, “100” or “10Prod”

  For a complete list of member names that must be enclosed in quotation marks, see “Understanding the Rules for Naming Dimensions and Members” on page 143.

- End each IF statement in a formula with an ENDIF statement.

  For example, the following formula contains a simple IF... ENDIF statement. You can apply this formula to the Commission member in a database outline:

  ```
  IF(Sales < 100)
     Commission = 0;
  ENDIF;
  ```

If you are using an IF statement nested within another IF statement, end each IF with an ENDIF, as illustrated in the following example:

```
"Opening Inventory"
(IF (@ISMBR(Budget))
   IF (@ISMBR(Jan))
   "Opening Inventory" = Jan;
   ELSE
   "Opening Inventory" = @PRIOR("Ending Inventory");
   ENDIF;
ENDIF;)
```

- You do not need to end ELSE or ELSEIF statements with ENDIFs, as illustrated in the following example:

  ```
  IF (@ISMBR(@DESCENDANTS(West)) OR
  @ISMBR(@DESCENDANTS(East)
     Marketing = Marketing * 1.5;
  ELSEIF(@ISMBR(@DESCENDANTS(South)))
     Marketing = Marketing * .9;
  ELSE Marketing = Marketing * 1.1;
  ENDIF;
  ```

  Note: If you use ELSE IF (with a space in between) rather than ELSEIF (one word) in a formula, you must supply an ENDIF for the IF statement.

- Although ending ENDIF statements with a semicolon (;) is not required, it is good practice to follow each ENDIF statement in a formula with a semicolon.

When writing formulas, you can check the syntax using the Formula Editor syntax checker. For a comprehensive discussion, including examples, of the main types of formulas, see “Checking Formula Syntax” on page 506.

For detailed information on syntax for Analytic Services functions and commands, see the *Technical Reference*.

Reviewing the Process for Creating Formulas

You use Formula Editor to create formulas. Formula Editor is a tab in the Member Properties dialog box in Outline Editor. You can type the formulas directly into the formula text area, or you can use the Formula Editor user interface features to create the formula.

Formulas are plain text. If required, you can create a formula in the text editor of your choice and paste it into Formula Editor.

To create a formula, follow this process:

1. In Outline Editor, select the member to which to apply the formula.
2. Open Formula Editor.

   For more information, see "Creating and Editing Formulas in Outlines" in the *Essbase Administration Services Online Help*.
3. Enter the formula text.

   For more information on entering the formula text in the Formula Editor, see "Creating and Editing Formulas in Outlines" in the *Essbase Administration Services Online Help*. For more information about composing the formula itself, see "Composing Formulas" on page 483.
4. Check the formula syntax.

   For more information, see "Checking Formula Syntax" on page 506.
5. Save the formula.

   For more information, see "Creating and Editing Formulas in Outlines" in the *Essbase Administration Services Online Help*.
6. Save the outline.

   For more information, see "Saving Outlines" in the *Essbase Administration Services Online Help*.

Displaying Formulas

➤ To display an existing formula, use any of the following methods:

Tool	Topic	Location
Administration Services	Creating and Editing Formulas in Outlines	*Essbase Administration Services Online Help*
ESSCMD	GETMBRCALC	*Technical Reference*
MaxL	**query database**	*Technical Reference*

Composing Formulas

The following sections discuss and give examples of the main types of formulas:

For detailed examples of formulas, see Chapter 23, "Reviewing Examples of Formulas."

Before writing formulas, review the guidelines in "Understanding Formula Syntax" on page 480.

Basic Equations

You can apply a mathematical operation to a formula to create a basic equation. For example, you can apply the following formula to the Margin member in Sample Basic.

```
Sales - COGS;
```

In a calculation script, you define basic equations as follows:

```
Member = mathematical operation;
```

where *Member* is a member name from the database outline and *mathematical operation* is any valid mathematical operation, as illustrated in the following example:

```
Margin = Sales - COGS;
```

Whether the example equation is in the database outline or in a calculation script, Analytic Services cycles through the database subtracting the values in COGS from the values in Sales and placing the results in Margin.

As another example, you can apply the following formula to a Markup member:

```
(Retail - Cost) % Retail;
```

In a calculation script, this formula is as follows:

```
Markup = (Retail - Cost) % Retail;
```

In this example, Analytic Services cycles through the database subtracting the values in Cost from the values in Retail, calculating the resulting values as a percentage of the values in Retail, and placing the result in Markup.

For an explanation of the nature of multidimensional calculations, see "About Multidimensional Calculation Concepts" on page 466

Conditional Tests

You can define formulas that use a conditional test or a series of conditional tests to control the flow of calculation.

The IF and ENDIF commands define a *conditional block*. The formulas between the IF and the ENDIF commands are executed only if the test returns TRUE (1). You can use the ELSE and ELSEIF commands to specify alternative actions if the test returns FALSE (0). The formulas following each ELSE command are executed only if the previous test returns FALSE (0). Conditions following each ELSEIF command are tested only if the previous IF command returns FALSE (0).

For information about and examples of the syntax of the IF and ENDIF commands, see "Understanding Formula Syntax" on page 480.

When you use a conditional formula in a calculation script, you must enclose it in parentheses and associate it with a member in the database outline, as shown in the examples in this section.

In conjunction with an IF command, you can use functions that return TRUE or FALSE (1 or 0, respectively) based on the result of a conditional test. These functions are known as *Boolean functions*.

You use Boolean functions to determine which formula to use. The decision is based on the characteristics of the current member combination. For example, you might want to restrict a certain calculation to the members in the Product dimension that contain input data. In this case, you preface the calculation with an IF test based on @ISLEV(Product,0).

If one of the function parameters is a cross-dimensional member, such as @ISMBR(Sales -> Budget), all of the parts of the cross-dimensional member must match the properties of the current cell to return a value of TRUE (1).

You can use the following Boolean functions to specify conditions.

Information You Need To Find	Use This Function
The current member has a specified accounts tag (for example, an Expense tag)	@ISACCTYPE
The current member is an ancestor of the specified member	@ISANCEST
The current member is an ancestor of the specified member, or the specified member itself	@ISIANCEST
The current member is a child of the specified member	@ISCHILD
The current member is a child of the specified member, or the specified member itself	@ISICHILD
The current member is a descendant of the specified member	@ISDESC
The current member is a descendant of the specified member, or the specified member itself	@ISIDESC
The current member of the specified dimension is in the generation specified	@ISGEN
The current member of the specified dimension is in the level specified	@ISLEV
The current member matches any of the specified members	@ISMBR
The current member is the parent of the specified member	@ISPARENT
The current member is the parent of the specified member, or the specified member itself	@ISIPARENT
The current member (of the same dimension as the specified member) is in the same generation as the specified member	@ISSAMEGEN
The current member (of the same dimension as the specified member) is in the same level as the specified member	@ISSAMELEV
The current member is a sibling of the specified member	@ISSIBLING
The current member is a sibling of the specified member, or the specified member itself	@ISISIBLING
A specified UDA (user-defined attribute) exists for the current member of the specified dimension	@ISUDA

When you place formulas on the database outline, you can use only the IF, ELSE, ELSEIF, and ENDIF commands and Boolean functions to control the flow of the calculations. You can use additional control commands in a calculation script.

For information about how to develop calculation scripts and how to use them to control how Analytic Services calculates a database, see Chapter 27, "Developing Calculation Scripts." For information on individual Analytic Services functions and calculation commands, see the *Technical Reference*.

Examples of Conditional Tests

You can apply the following formula to a Commission member in the database outline. In the first example, the formula calculates commission at 1% of sales if the sales are greater than 500000:

```
IF(Sales > 500000)
Commission = Sales * .01;
ENDIF;
```

If you place the formula in a calculation script, you need to associate the formula with the Commission member as follows:

```
Commission(IF(Sales > 500000)
Commission = Sales * .01;
ENDIF;)
```

Analytic Services cycles through the database, performing these calculations:

1. The IF statement checks to see if the value of Sales for the current member combination is greater than 500000.
2. If Sales is greater than 500000, Analytic Services multiplies the value in Sales by 0.01 and places the result in Commission.

In the next example, the formula tests the ancestry of the current member and then applies the appropriate Payroll calculation formula.

```
IF(@ISIDESC(East) OR @ISIDESC(West))
Payroll = Sales * .15;
ELSEIF(@ISIDESC(Central))
Payroll = Sales * .11;
ELSE
Payroll = Sales * .10;
ENDIF;
```

If you place the formula in a calculation script, you need to associate the formula with the Payroll member as follows:

```
Payroll(IF(@ISIDESC(East) OR @ISIDESC(West))
Payroll = Sales * .15;
ELSEIF(@ISIDESC(Central))
Payroll = Sales * .11;
ELSE
Payroll = Sales * .10;
ENDIF;)
```

Analytic Services cycles through the database, performing the following calculations:

1. The IF statement uses the @ISIDESC function to check if the current member on the Market dimension is a descendant of either East or West.
2. If the current member on the Market dimension is a descendant of East or West, Analytic Services multiplies the value in Sales by 0.15 and moves on to the next member combination.
3. If the current member is not a descendant of East or West, the ELSEIF statement uses the @ISIDESC function to check if the current member is a descendant of Central.
4. If the current member on the Market dimension is a descendant of Central, Analytic Services multiplies the value in Sales by 0.11 and moves on to the next member combination.
5. If the current member is not a descendant of East, West, or Central, Analytic Services multiplies the value in Sales by 0.10 and moves on to the next member combination.

For information on the nature of multidimensional calculations, see “About Multidimensional Calculation Concepts” on page 466. For information on the @ISIDESC function, see the *Technical Reference*.

Value-Related Formulas

Use this section to find information about formulas related to values:

Using Interdependent Values

Analytic Services optimizes calculation performance by calculating formulas for a range of members in the same dimension at the same time. However, some formulas require values from members of the same dimension, and Analytic Services may not yet have calculated the required values.

A good example is that of cash flow, in which the opening inventory is dependent on the ending inventory from the previous month.

In Sample Basic, the Opening Inventory and Ending Inventory values need to be calculated on a month-by-month basis.

	Jan	Feb	Mar
Opening Inventory	100	120	110
Sales	50	70	100
Addition	70	60	150
Ending Inventory	120	110	160

Assuming that the Opening Inventory value for January is loaded into the database, the required calculation is as follows:

```
1. January Ending   = January Opening – Sales + Additions
2. February Opening = January Ending
3. February Ending  = February Opening – Sales + Additions
4. March Opening    = February Ending
5. March Ending     = March Opening – Sales + Additions
```

You can calculate the required results by applying interdependent, multiple equations to a single member in the database outline.

The following formula, applied to the Opening Inventory member in the database outline, calculates the correct values:

```
IF(NOT @ISMBR (Jan))
    "Opening Inventory" = @PRIOR("Ending Inventory");
ENDIF;
"Ending Inventory" = "Opening Inventory" - Sales + Additions;
```

If you place the formula in a calculation script, you need to associate the formula with the Opening Inventory member as follows:

```
"Opening Inventory" (IF(NOT @ISMBR (Jan))
"Opening Inventory" = @PRIOR("Ending Inventory");
ENDIF;
"Ending Inventory" = "Opening Inventory" - Sales + Additions;)
```

Analytic Services cycles through the months, performing the following calculations:

1. The IF statement and @ISMBR function check that the current member on the Year dimension is not Jan. This step is necessary because the Opening Inventory value for Jan is an input value.
2. If the current month is not Jan, the @PRIOR function obtains the value for the Ending Inventory for the previous month. This value is then allocated to the Opening Inventory of the current month.
3. The Ending Inventory is calculated for the current month.

Note: To calculate the correct results, it is necessary to place the above formula on a single member, Opening Inventory. If you place the formulas for Opening Inventory and Ending Inventory on their separate members, Analytic Services calculates Opening Inventory for all months and then Ending Inventory for all months. This organization means that the value of the Ending Inventory of the previous month is not available when Opening Inventory is calculated.

Calculating Variances or Percentage Variances Between Actual and Budget Values

You can use the @VAR and @VARPER functions to calculate a variance or percentage variance between budget and actual values.

You may want the variance to be positive or negative, depending on whether you are calculating variance for members on the accounts dimension that are expense or non-expense items:

- Expense items. You want Analytic Services to show a positive variance if the actual values are lower than the budget values. For example, you want Analytic Services to show a positive variance if actual costs are lower than budgeted costs.
- Non-expense items. You want Analytic Services to show a negative variance if the actual values are lower than the budget values. For example, you want Analytic Services to show a negative variance if actual sales are lower than budgeted sales.

By default, Analytic Services assumes that members are non-expense items and calculates the variance accordingly.

➤ To tell Analytic Services that a member is an expense item, use this procedure:

1. In Outline Editor, select the member.

   The member must be on the dimension tagged as accounts.

2. Open Formula Editor.

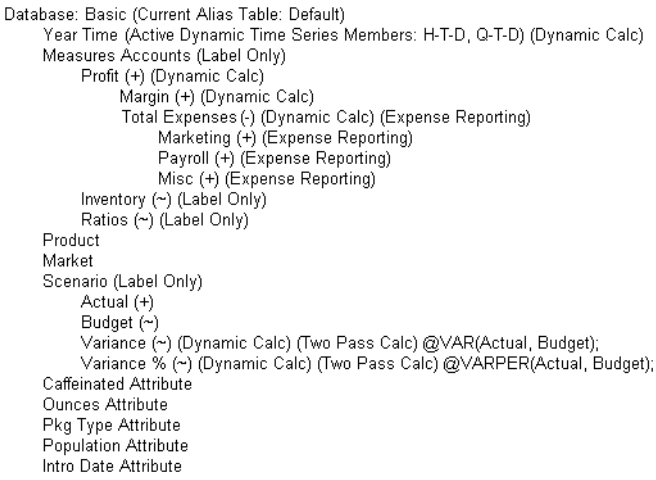
   See “Creating and Editing Formulas in Outlines” in the *Essbase Administration Services Online Help*.

3. Tag the member as an expense item. See “Setting Variance Reporting Properties” on page 158.

   When you use the @VAR or @VARPER functions, Analytic Services shows a positive variance if the actual values are lower than the budget values.

For example, in Sample Basic, the children of Total Expenses are expense items. The Variance and Variance % members of the Scenario dimension calculate the variance between the Actual and Budget values.

Figure 174: Sample Basic Showing Expense Items



Allocating Values

You can allocate values that are input at the parent level across child members in the same dimension or in different dimensions by using the following allocation functions.

Allocated Values	Function To Use
Values from a member, cross-dimensional member, or value across a member list within the same dimension. The allocation is based on a variety of specified criteria.	@ALLOCATE
Values from a member, cross-dimensional member, or value across multiple dimensions. The allocation is based on a variety of specified criteria.	@MDALLOCATE

Note: For examples of calculation scripts using the @ALLOCATE and @MDALLOCATE functions, see “Allocating Values Within or Across Dimensions” on page 616 and the *Technical Reference*.

Forecasting Values

You can manipulate data for the purposes of smoothing data, interpolating data, or calculating future values by using the following forecasting functions.

Data Manipulation	Function To Use
To apply a moving average to a data set and replace each term in the list with a trailing average. This function modifies the data set for smoothing purposes.	@MOVAVG
To apply a moving maximum to a data set and replace each term in the list with a trailing maximum. This function modifies the data set for smoothing purposes.	@MOVMAX
To apply a moving median to a data set and replace each term in the list with a trailing median. This function modifies the data set for smoothing purposes.	@MOVMED
To apply a moving minimum to a data set and replace each term in the list with a trailing minimum. This function modifies the data set for smoothing purposes.	@MOVMIN
To apply a moving sum to a data set and replace each term with a trailing sum. This function modifies the data set for smoothing purposes.	@MOVSUM
To apply a moving sum to a data set and replace each term with a trailing sum. Specify how to assign values to members before you reach the number to sum. This function modifies the data set for smoothing purposes.	@MOVSUMX
To apply a smoothing spline to a set of data points. A spline is a mathematical curve that is used to smooth or interpolate data.	@SPLINE
To calculate future values and base the calculation on curve-fitting to historical values.	@TREND

For information about specific Analytic Services functions, see the *Technical Reference*.

Using Member Relationships to Look Up Values

You can use the member combination that Analytic Services is currently calculating to look up specific values. These functions are referred to as relationship functions.

Look-up Value	Function To Use
The ancestor values of the specified member combination	@ANCESTVAL
The numeric value of the attribute from the specified numeric or date attribute dimension associated with the current member	@ATTRIBUTEVAL
The text value of the attribute from the specified text attribute dimension associated with the current member	@ATTRIBUTESVAL
The value (TRUE or FALSE) of the attribute from the specified Boolean attribute dimension associated with the current member	@ATTRIBUTEBVAL
The generation number of the current member combination for the specified dimension	@CURGEN
The level number of the current member combination for the specified dimension	@CURLEV
The generation number of the specified member	@GEN
The level number of the specified member	@LEV
The ancestor values of the specified member combination across multiple dimensions	@MDANCESTVAL
The shared ancestor values of the specified member combination	@SANCESTVAL
The parent values of the specified member combination	@PARENTVAL
The parent values of the specified member combination across multiple dimensions	@MDPARENTVAL
The shared parent values of the specified member combination	@SPARENTVAL
A data value from another database to be used for calculation of a value from the current database	@XREF

For information about specific Analytic Services functions, see the *Technical Reference*.

Using Substitution Variables

Substitution variables act as placeholders for information that changes regularly; for example, time period information. You can use substitution variables in formulas that you include in a calculation script. You cannot use substitution variables in formulas that you apply to the database outline.

When you run a calculation script, Analytic Services replaces the substitution variable with the value you have assigned to it. You can create and assign values to substitution variables using Essbase Administration Services or ESSCMD.

You can set substitution variables at the server, application, and database levels. Analytic Services must be able to access the substitution variable from the application and database on which you are running the calculation script.

For information on creating and assigning values to substitution variables, see "Using Substitution Variables" on page 133.

➤ To use a substitution variable in a calculation script, type an ampersand (&) followed by the substitution variable name.

Analytic Services treats any text string preceded by & as a substitution variable.

For example, assume that the substitution variable UpToCurr is defined as Jan:Jun. You can use the following @ISMBR function as part of a conditional test in a calculation script:

```
@ISMBR(&UpToCurr)
```

Before Analytic Services runs the calculation script, it replaces the substitution variable, as follows:

```
@ISMBR(Jan:Jun)
```

Member-Related Formulas

This section provides information you need to create formulas that refer to members:

- "Specifying Member Lists and Ranges" on page 495
- "Generating Member Lists" on page 496

- "Manipulating Member Names" on page 499
- "Working with Member Combinations across Dimensions" on page 499

Specifying Member Lists and Ranges

In some functions you may need to specify more than one member, or you may need to specify a range of members. For example, the @ISMBR function tests to see if a member that is currently being calculated matches any of a list or range of specified members. You can specify members using the following syntax:

Member List or Range	Syntax
A single member	The member name. For example: Mar2001
A list of members	A comma-delimited (,) list of member names. For example: Mar2001, Apr2001, May2001
A range of all members at the same level, between and including the two defining members	The two defining member names separated by a colon (:). For example: Jan2000:Dec2000
A range of all members in the same generation, between and including the two defining members	The two defining member names separated by two colons (::). For example: Q1_2000::Q4_2000
A function-generated list of members or a range of members	For a list of member list contents and corresponding functions, see "Generating Member Lists" on page 496.
A combination of ranges and list	Separate each range, list, and function with a comma (,). For example: Q1_97::Q4_98, FY99, FY2000 or @SIBLINGS(Dept01), Dept65:Dept73, Total_Dept

If you do not specify a list of members or a range of members in a function that requires either, Analytic Services uses the level 0 members of the dimension tagged as time. If no dimension is tagged as time, Analytic Services displays an error message.

Generating Member Lists

You can generate member lists that are based on a specified member by using the these member set functions.

Contents of Member List	Function
All ancestors of the specified member, including ancestors of the specified member as a shared member. This function does not include the specified member.	@ALLANCESTORS
All ancestors of the specified member, including ancestors of the specified member as a shared member. This function includes the specified member.	@IALLANCESTORS
The ancestor of the specified member at the specified generation or level.	@ANCEST
All ancestors of the specified member (optionally up to the specified generation or level) but not the specified member.	@ANCESTORS
All ancestors of the specified member (optionally up to the specified generation or level) including the specified member.	@IANCESTORS
All children of the specified member, but not including the specified member.	@CHILDREN
All children of the specified member, including the specified member.	@ICHILDREN
The current member being calculated for the specified dimension.	@CURRMBR
All descendants of the specified member (optionally up to the specified generation or level), but not the specified member nor descendants of shared members.	@DESCENDANTS
All descendants of the specified member (optionally up to the specified generation or level), including the specified member, but not descendants of shared members.	@IDESCENDANTS

Contents of Member List	Function
All descendants of the specified member (optionally up to the specified generation or level), including descendants of shared members, but not the specified member.	@RDESCENDANTS
All descendants of the specified member (optionally up to the specified generation or level), including the specified member and descendants of shared members.	@IRDESCENDANTS
All members of the specified generation in the specified dimension.	@GENMBRS
All members of the specified level in the specified dimension.	@LEVMBRS
All siblings of the specified member, but not the specified member.	@SIBLINGS
All siblings of the specified member, including the specified member.	@ISIBLINGS
All siblings that precede the specified member in the database outline, but not the specified member.	@LSIBLINGS
All siblings that follow the specified member in the database outline, but not the specified member.	@RSIBLINGS
All siblings that precede the specified member in the database outline, including the specified member.	@ILSIBLINGS
All siblings that follow the specified member in the database outline, including the specified member.	@IRSIBLINGS
Separate lists of members to be processed by functions that require multiple list arguments.	@LIST
The member with the name that is provided as a character string.	@MEMBER
A merged list of two member lists to be processed by another function.	@MERGE
A member list that crosses the specified member from one dimension with the specified member range from another dimension.	@RANGE

Contents of Member List	Function
A member list the identifies all shared members among the specified members.	@SHARE
A member list that identifies the range of members using the level of the arguments, determining the cross product of all members in the range, and pruning the set to include only the range requested.	@XRANGE
A list of members from which some members have been removed.	@REMOVE
All members that match the specified wildcard selection.	@MATCH
The parent of the current member being calculated in the specified dimension.	@PARENT
All members of the specified generation or level that are above or below the specified member.	@RELATIVE
All members that have a common (UDA) user-defined attribute defined on Analytic Server.	@UDA
All base-dimension members that are associated with the specified attribute-dimension member.	@ATTRIBUTE
All base members that are associated with attributes that satisfy the specified conditions.	@WITHATTR

For information about specific Analytic Services functions, see the *Technical Reference*.

Manipulating Member Names

You can work with member names as character strings by using the following functions:

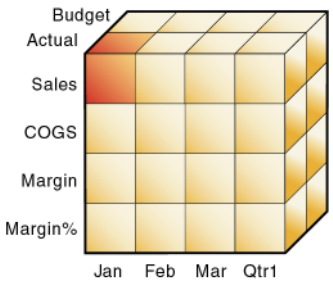
Character String Manipulation	Function To Use
To create a character string that is the result of appending a member name or specified character string to another member name or character string	@CONCATENATE
To return a member name as a string	@NAME
To return a substring of characters from another character string or from a member name	@SUBSTRING

Working with Member Combinations across Dimensions

Use the cross-dimensional operator to point to data values of specific member combinations. Create the cross-dimensional operator using a hyphen (-) and a greater than symbol (>). Do not leave spaces in between the cross-dimensional operator and the member names.

For example, in this simplified illustration, the shaded data value is Sales -> Jan -> Actual.

Figure 175: Defining a Single Data Value by Using the Cross-Dimensional Operator



The following example illustrates how to use the cross-dimensional operator. This example allocates miscellaneous expenses to each product in each market.

The value of Misc_Expenses for all products in all markets is known. The formula allocates a percentage of the total Misc_Expenses value to each Product -> Market combination. The allocation is based on the value of Sales for each product in each market.

```
Misc_Expenses = Misc_Expenses -> Market -> Product *
(Sales / ( Sales -> Market -> Product));
```

Analytic Services cycles through the database, performing these calculations:

1. Analytic Services divides the Sales value for the current member combination by the total Sales value for all markets and all products (Sales -> Market -> Product).
2. It multiplies the value calculated in step 1 by the Misc_Expenses value for all markets and all products (Misc_Expenses -> Market -> Product).
3. It allocates the result to Misc_Expenses for the current member combination.

Consider carefully how you use the cross-dimensional operator, as it can have significant performance implications. For information about optimizing and the cross-dimensional operator, see “Using Cross-Dimensional Operators (->)” on page 1198.

Formulas That Use Various Types of Functions

Use this section to find information about formulas that use other types of formulas:

Mathematical Operations

You can perform many mathematical operations in formulas by using the following mathematical functions.

Operation	Function
To return the absolute value of an expression	@ABS
To return the average value of the values in the specified member list	@AVG
To return the value of e (the base of natural logarithms) raised to power of the specified expression	@EXP
To return the factorial of an expression	@FACTORIAL
To return the next lowest integer value of a member or expression	@INT
To return the natural logarithm of a specified expression	@LN
To return the logarithm to a specified base of a specified expression	@LOG
To return the base-10 logarithm of a specified expression	@LOG10
To return the maximum value among the expressions in the specified member list	@MAX
To return the maximum value among the expressions in the specified member list, with the ability to skip zero and #MISSING values	@MAXS
To return the minimum value among the expressions in the specified member list	@MIN
To return the minimum value among the expressions in the specified member list, with the ability to skip zero and #MISSING values	@MINS
To return the modulus produced by the division of two specified members	@MOD
To return the value of the specified member raised to the specified power	@POWER
To return the remainder value of an expression	@REMAINDER

Operation	Function
To return the member or expression rounded to the specified number of decimal places	@ROUND
To return the summation of values of all specified members	@SUM
To return the truncated value of an expression	@TRUNCATE
To return the variance (difference) between two specified members. See "Calculating Variances or Percentage Variances Between Actual and Budget Values" on page 490.	@VAR
To return the percentage variance (difference) between two specified members. See "Calculating Variances or Percentage Variances Between Actual and Budget Values" on page 490.	@VARPER

For information about specific Analytic Services functions, see the *Technical Reference*.

Statistical Functions

You can use these statistical functions to calculate advanced statistics in Analytic Services.

Calculated Value	Function to Use
The correlation coefficient between two parallel data sets	@CORRELATION
The number of values in the specified data set	@COUNT
The median, or middle number, in the specified data set	@MEDIAN
The mode, or the most frequently occurring value, in the specified data set	@MODE
The rank of the specified member or value in the specified data set	@RANK
The standard deviation, based upon a sample, of the specified members	@STDEV
The standard deviation, based upon the entire population, of the specified members	@STDEVP

Calculated Value	Function to Use
The standard deviation, crossed with a range of members, of the specified members	@STDEVRANGE
The variance, based upon a sample, of the specified data set	@VARIANCE
The variance, based upon the entire population, of the specified data set	@VARIANCEP

For information about specific Analytic Services functions, see the *Technical Reference*.

Range Functions

You can execute a function for a range of members by using these range functions.

Calculation	Function to Use
The average value of a member across a range of members	@AVGRANGE
A range of members that is based on the relative position of the member combination Analytic Services is currently calculating.	@CURRMBRRANGE
The maximum value of a member across a range of members	@MAXRANGE
The maximum value of a member across a range of members, with the ability to skip zero and #MISSING values	@MAXSRANGE
The next or *n*th member in a range of members, retaining all other members identical to the current member across multiple dimensions	@MDSHIFT
The minimum value of a member across a range of members	@MINRANGE
The minimum value of a member across a range of members, with the ability to skip zero and #MISSING values	@MINSRANGE
The next or *n*th member in a range of members.	@NEXT

Calculation	Function to Use
The next or *n* th member in a range of members, with the option to skip #MISSING, zero, or both values.	@NEXTS
The previous or *n* th previous member in a range of members	@PRIOR
The previous or *n* th previous member in a range of members, with the option to skip #MISSING, zero, or both values.	@PRIORS
The next or *n* th member in a range of members, retaining all other members identical to the current member and in the specified dimension	@SHIFT In some cases, @SHIFTPLUS or @SHIFTMINUS.
The summation of values of all specified members across a range of members	@SUMRANGE

For information about specific Analytic Services functions, see the *Technical Reference*.

Financial Functions

You can include financial calculations in formulas by using these financial functions.

Calculation	Function To Use
An accumulation of values up to the specified member	@ACCUM
The proceeds of a compound interest calculation	@COMPOUND
A series of values that represent the compound growth of the specified member across a range of members	@COMPOUNDGROWTH
Depreciation for a specific period, calculated using the declining balance method.	@DECLINE

Calculation	Function To Use
A value discounted by the specified rate, from the first period of the range to the period in which the amount to discount is found	@DISCOUNT
A series of values that represents the linear growth of the specified value	@GROWTH
The simple interest for a specified member at a specified rate	@INTEREST
The internal rate of return on a cash flow	@IRR
The Net Present Value of an investment (based on a series of payments and incomes)	@NPV
The period-to-date values of members in the dimension tagged as time	@PTD
The amount per period that an asset in the current period may be depreciated (calculated across a range of periods). The depreciation method used is straight-line depreciation.	@SLN
The amount per period that an asset in the current period may be depreciated (calculated across a range of periods). The depreciation method used is sum of the year's digits.	@SYD

For information about specific Analytic Services functions, see the *Technical Reference*.

Date and Time Functions

You can use dates with other functions by using this date function.

Date Conversion	Function To Use
Convert date strings to numbers that can be used in calculation formulas	@TODATE

Calculation Mode Functions

You can specify which calculation mode that Analytic Services uses to calculate a formula by using @CALCMODE.

Specification	Function To Use
To specify that Analytic Services uses cell, block, bottom-up, and top-down calculation modes to calculate a formula.	@CALCMODE

Note: You can also use the configuration setting CALCMODE to set calculation modes to BLOCK or BOTTOMUP at the database, application, or server level. For details, see the *Technical Reference*, under "`essbase.cfg` Settings" for CALCMODE or "Analytic Services Functions" for @CALCMODE.

Custom-Defined Functions

Custom-defined functions are calculation functions that you create to perform calculations not otherwise supported by the Analytic Services calculation scripting language. You can use custom-defined functions in formulas and calculation scripts. These custom-developed functions are written in the Java programming language and registered on the Analytic Server. The Analytic Services calculator framework calls them as external functions.

Custom-defined functions are displayed in the functions tree in Calculation Script Editor. From this tree, you can select a custom-defined function to insert into a formula.

For a detailed explanation of how to develop and use custom-defined functions, see Chapter 30, "Developing Custom-Defined Calculation Functions."

Checking Formula Syntax

Analytic Services includes Analytic Server-based formula syntax checking that tells you about syntax errors in formulas. For example, Analytic Services tells you if you have mistyped a function name. Unknown names can be validated against a list of custom-defined macro and function names. If you are not connected to a server or the application associated with the outline, Analytic Services may connect you to validate unknown names.

A syntax checker cannot tell you about semantic errors in a formula. Semantic errors occur when a formula does not work as you expect. To find semantic errors, run the calculation and check the results to ensure that they are as you expect.

Analytic Services displays the syntax checker results at the bottom of the Formula Editor. If Analytic Services finds no syntax errors, it displays the "No errors" message.

If Analytic Services finds one or more syntax errors, it displays the number of the line that includes the error and a brief description of the error. For example, if you do not include a semicolon end-of-line character at the end of a formula, Analytic Services displays a message similar to the message shown in Figure 176.

Figure 176: Formula Editor Syntax Checker, Syntax Error Message

```
Error: line 1: invalid statement; expected semicolon
```

If a formula passes validation in Formula Editor or Outline Editor, but Analytic Server detects semantic errors when the outline is saved, check the following:

- The incorrect formula is saved as part of the outline, even though it contains errors.
- Analytic Server writes a message in the application log that indicates what the error is and displays the incorrect formula.
- Analytic Server writes an error message to the comment field of the member associated with the incorrect formula. The message indicates that the incorrect formula was not loaded. You can view this comment in Outline Editor by closing and reopening the outline.
- If you do not correct the member formula, and a calculation that includes that member is run, the formula is ignored during the calculation.

After you have corrected the formula and saved the outline, the message in the member comment is deleted. You can view the updated comment when you reopen the outline.

➤ To check formula syntax, see "Creating and Editing Formulas in Outlines" in the *Essbase Administration Services Online Help*.

Estimating Disk Size for a Calculation

You can estimate the disk size required for a single CALC ALL given either a full data load or a partial data load. For more information, see "Estimating Calculation Affects on Database Size" on page 1180.

➤ To estimate disk size for a calculation, see ESTIMATEFULLDBSIZE in the *Technical Reference*.

Using Formulas in Partitions

An Analytic Services partition can span multiple Analytic Servers, processors, or computers. For a comprehensive discussion of partitioning, see Chapter 13, "Designing Partitioned Applications" and Chapter 14, "Creating and Maintaining Partitions".

You can use formulas in partitioning, just as you use formulas on your local database. However, if a formula you use in one database references a value from another database, Analytic Services has to retrieve the data from the other database when calculating the formula. In this case, you need to ensure that the referenced values are up-to-date and to consider carefully the performance impact on the overall database calculation. For a discussion of how various options affect performance, see "Writing Calculation Scripts for Partitions" on page 602.

With transparent partitions, you need to consider carefully how you use formulas on the data target. For a detailed example of the relationship between member formulas and transparent partitioning, see "Transparent Partitions and Member Formulas" on page 255. For a discussion of the performance implications, see "Performance Considerations for Transparent Partition Calculations" on page 254.

Chapter 23

Reviewing Examples of Formulas

This chapter provides detailed examples of formulas, which you may want to adapt for your own use. For examples of using formulas in calculation scripts, see Chapter 28, "Reviewing Examples of Calculation Scripts."

The information in this chapter does not apply to aggregate storage outlines. For information about developing formulas in MDX for aggregate storage outline members, see "Developing Formulas on Aggregate Storage Outlines" on page 1309.

This chapter includes the following sections:

Calculating Period-to-Date Values

If the outline includes a dimension tagged as accounts, you can use the @PTD function to calculate period-to-date values. You can also use Dynamic Time Series members to calculate period-to-date values. For an explanation of how to calculate time series data, see Chapter 26, "Calculating Time Series Data."

For example, the following figure shows the Inventory branch of the Measures dimension from the Sample Basic database.

Figure 177: Inventory Branch from Sample Basic Outline

```
Inventory (~) (Label Only)
   Opening Inventory (+) (TB First) (Expense Reporting) IF(NOT @ISMBR(Jan))
   Additions (~) (Expense Reporting)
   Ending Inventory (~) (TB Last) (Expense Reporting)
```

To calculate period-to-date values for the year and for the current quarter, add two members to the Year dimension, QTD for quarter-to-date and YTD for year-to-date:

```
QTD (~) @PTD(Apr:May)
YTD (~) @PTD(Jan:May);
```

For example, assuming that the current month is May, you would add this formula to the QTD member:

```
@PTD(Apr:May);
```

And you would add this formula on the YTD member:

```
@PTD(Jan:May);
```

Analytic Services sums the values for the range of months as appropriate. However, Opening Inventory has a time balance tag, First, and Ending Inventory has a time balance tag, Last. Analytic Services takes these values and treats them accordingly. For more information on time balance tags, see "Calculating First, Last, and Average Values" on page 567.

The following table provides an example of the calculation results for the members in the Inventory branch and for the Sales member:

Measures -> Time	Jan	Feb	Mar	Apr	May	QTD	YTD
Opening Inventory	100	110	120	110	140	**110**	**100**
Additions	110	120	100	160	180	**340**	**670**
Sales	100	110	110	130	190	**320**	**640**
Ending Inventory	110	120	110	140	130	**130**	**130**

The values for Sales and Additions have been summed.

Opening Inventory has a First tag. For QTD, Analytic Services takes the first value in the current quarter, which is Apr. For YTD, Analytic Services takes the first value in the year, which is Jan.

Ending Inventory has a Last tag. For QTD, Analytic Services takes the last value in the current quarter, which is May. For YTD, Analytic Services takes the last value in the year, which is also May.

Calculating Rolling Values

You can use the @AVGRANGE function to calculate rolling averages and the @ACCUM function to calculate rolling year-to-date values.

For example, assume that a database contains monthly Sales data values and that the database outline includes the members AVG_Sales and YTD_Sales.

You would add this formula to the AVG_Sales member:

```
@AVGRANGE(SKIPNONE, Sales, @CURRMBRRANGE(Year, LEV, 0, , 0));
```

And you would add this formula on the YTD_Sales member:

```
@ACCUM(Sales);
```

Analytic Services calculates the average Sales values across the months in the dimension tagged as time. The SKIPNONE parameter means that all values are included, even #MISSING values. Analytic Services places the results in AVG_Sales. For an explanation of how Analytic Services calculates #MISSING values, see “Consolidating #MISSING Values” on page 1217.

This table shows the results when Analytic Services calculates the cumulative Sales values and places the results in YTD_Sales:

Measures -> Time	Jan	Feb	Mar	Qtr1
Sales	100	200	300	**600**
AVG_Sales	100	150	200	#MISSING
YTD_Sales	100	300	600	#MISSING

The values for AVG_Sales are averages of the months-to-date. For example, AVG_Sales -> Mar is an average of Sales for Jan, Feb, and Mar.

The values for YTD_Sales are the cumulative values up to the current month. So YTD_Sales -> Feb is the sum of Sales -> Jan and Sales -> Feb.

Calculating Monthly Asset Movements

You can use the @PRIOR function to calculate values based on a previous month's value.

For example, assume that a database contains assets data values that are stored on a month-by-month basis. You can calculate the difference between the assets values of successive months (the asset movement) by subtracting the previous month's value from the present month's value.

Assume these three members manage the asset values for the database:

- Assets for the monthly asset values
- Asset_MVNT for the asset movement values
- Opening_Balance for the asset value at the beginning of the year

For Jan, the Asset_MVNT value is calculated by subtracting the Opening_Balance value from the Jan value.

You would add this formula on the Asset_MVNT member:

```
IF(@ISMBR(Jan)) Asset_MVNT = Assets - Opening_Balance;
ELSE Asset_MVNT = Assets - @PRIOR(Assets);
ENDIF;
```

This table shows the results when Analytic Services calculates the difference between the values of assets in successive months:

Assets -> Time	Opening_Balance	Jan	Feb	Mar
Assets	1200	1400	1300	1800
Asset_MVNT		**200**	**-100**	**500**

Analytic Services cycles through the months, performing these calculations:

1. The IF statement and @ISMBR function check to see if the current member on the Year dimension is Jan. This check is necessary because the Asset_MVNT value for Jan cannot be calculated by subtracting the previous month's value.
2. If the current member on the Year dimension is Jan, Analytic Services subtracts the Opening_Balance from the Jan -> Assets value and places the result in Jan -> Asset_MVNT.
3. If the current member on the Year dimension is not Jan, the @PRIOR function obtains the value for the previous month's assets. Analytic Services subtracts the previous month's assets from the current month's assets. It places the result in the current month's Asset_MVNT value.

Testing for #MISSING Values

You can test for #MISSING values in a database. For an explanation of how Analytic Services calculates #MISSING values, see "Consolidating #MISSING Values" on page 1217.

Assume that a database outline contains a member called Commission. Commission is paid at 10% of sales when the Sales value for the current member combination is not #MISSING. When applied to a Commission member in the database outline, the following formula calculates Commission:

```
IF(Sales <> #MISSING) Commission = Sales * .1;
ELSE Commission = #MISSING;
ENDIF;
```

If you place the formula in a calculation script, you need to associate it with the commission member as follows:

```
Commission(IF(Sales <> #MISSING) Commission = Sales * .1;
ELSE Commission = #MISSING;
ENDIF;);
```

Analytic Services cycles through the database, performing the following calculations:

1. The IF statement checks to see if the value of the Sales member for the current member combination is not #MISSING.
2. If Sales is not #MISSING, Analytic Services multiplies the value in the Sales member by 0.1 and places the result in the Commission member.
3. If Sales is #MISSING, Analytic Services places #MISSING in the Commission member.

Calculating an Attribute Formula

You can perform specific calculations on attribute-dimension members in a database.

Note: For a comprehensive discussion of attribute calculations, see "Calculating Attribute Data" on page 200.

For example, to calculate profitability by ounce for products sized in ounces, you can use the @ATTRIBUTEVAL function in a calculation formula. In the Sample Basic database, the Ratios branch of the Measures dimension contains a member called Profit per Ounce. The formula on this member is:

```
Profit/@ATTRIBUTEVAL(@NAME(Ounces));
```

Analytic Services cycles through the Products dimension, performing the following calculations:

1. For each base member that is associated with a member from the Ounces attribute dimension, the @ATTRIBUTEVAL function returns the numeric attribute value (for example, 12 for the member 12 under Ounces).

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

2. Analytic Services then divides Profit by the result of @ATTRIBUTEVAL to yield Profit per Ounce.

Note: For an explanation of the functions that you use to perform calculations on attributes in formulas, see "Using Attributes in Calculation Formulas" on page 206. For more information about the @ATTRIBUTEVAL function, see the *Technical Reference*.

Chapter 24

Defining Calculation Order

This chapter describes the order in which Analytic Services calculates a block storage database. You should understand the concepts of data blocks and of sparse and dense dimensions before using this information. For a review of this information, see "Sparse and Dense Dimensions" on page 62. You should also understand the use of levels and generations. For a review of this information, see "Generations and Levels" on page 36. If you use dynamic calculations, see "Understanding How Dynamic Calculation Changes Calculation Order" on page 553 for information on the calculation order for the dynamically calculated values.

Calculation order does not apply to aggregate storage databases.

This chapter includes these sections:

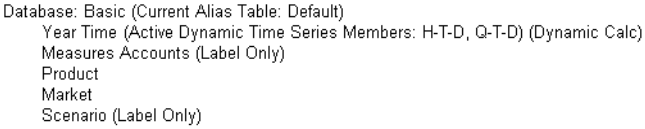
Data Storage in Data Blocks

Analytic Services stores data values in data blocks. Analytic Services creates a data block for each unique combination of sparse dimension members (providing that at least one data value exists for the combination).

Each data block contains all the dense dimension member values for its unique combination of sparse dimension members.

In the Sample Basic database, the Year, Measures, and Scenario dimensions are dense. The Product and Market dimensions are sparse.

Figure 178: Dimensions from the Sample Basic Database



Note: Sample Basic also contains five attribute dimensions. These dimensions are sparse, Dynamic Calc, meaning that attribute data is not stored in the database. For a comprehensive discussion of attributes, see Chapter 10, “Working with Attributes”.

Analytic Services creates a data block for each unique combination of members in the Product and Market dimensions (providing that at least one data value exists for the combination). For example, it creates one data block for the combination of 100-10, New York. This data block contains all the Year, Measures, and Scenario values for 100-10, New York.

Figure 179: Product and Market Dimensions from the Sample Basic Database

In Analytic Services, member combinations are denoted by the cross-dimensional operator. The symbol for the cross-dimensional operator is -> (a hyphen followed by a greater than symbol). So 100-10, New York is written 100-10 -> New York.

You can categorize data blocks as follows:

- Input. These blocks are created by loading data to cells in a block. Input blocks can be created for (1) sparse, level 0 member combinations or (2) sparse, upper level member combinations, when at least one of the sparse members is a parent level member. Input blocks can be level 0 or upper level blocks.
- Noninput. These blocks are created through calculations. For example, in Sample Basic, the East -> Cola block is created during a sparse calculation process (that is, the block did not exist before calculation).
- Level 0. These blocks are created for sparse member combinations when all of the sparse members are level 0 members. For example, in Sample Basic, New York -> Cola is a level 0 block because New York and Cola are level 0 members of their respective sparse dimensions. Level 0 blocks can be input or noninput blocks; for example, a level 0 noninput block is created during an allocation process, where data is loaded at a parent level and then allocated down to level 0.
- Upper level. These blocks are created for sparse member combinations when at least one of the sparse members is a parent level member. Upper level blocks can be input or noninput blocks.

For more information on levels and generations, see "Generations and Levels" on page 36. For information on how Analytic Services stores data in data blocks, see "Data Blocks and the Index System" on page 64.

Member Calculation Order

Analytic Services calculates a database at the data block level, bringing one or more blocks into memory and calculating the required values within the block. Analytic Services calculates the blocks in order, according to their block numbers. The database outline tells Analytic Services how to order the blocks. Within each block, Analytic Services calculates the values in order according to the hierarchy in the database outline. Therefore, overall, Analytic Services calculates a database based on the database outline.

When you perform a default calculation (CALC ALL) on a database, Analytic Services calculates the dimensions in this order:

If both a dimension tagged as accounts and a dimension tagged as time exist, and if formulas are applied to members on the accounts dimension, Analytic Services calculates as follows:

1. The dimension tagged as accounts
2. The dimension tagged as time
3. Other dense dimensions (in the order they are displayed in the database outline)
4. Other sparse dimensions (in the order they are displayed in the database outline)

Otherwise, Analytic Services calculates in this order:

1. Dense dimensions (in the order they display in the database outline)
2. Sparse dimensions (in the order they display in the database outline)

Note: Attribute dimensions, which are not included in the database consolidation, do not affect calculation order. For a comprehensive discussion of attribute dimensions, see Chapter 10, "Working with Attributes".

In the Sample Basic database, the dimensions are calculated in this order: Measures, Year, Scenario, Product, and Market.

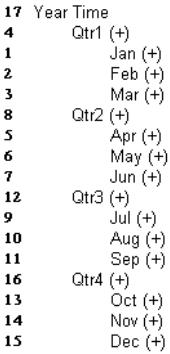
You can override the default order by using a calculation script. For a comprehensive discussion of how to develop and use calculation scripts, see Chapter 27, "Developing Calculation Scripts."

Understanding the Effects of Member Relationships

The order of calculation within each dimension depends on the relationships between members in the database outline. Within each branch of a dimension, level 0 values are calculated first followed by their level 1, parent value. Then the level 0 values of the next branch are calculated followed by their level 1, parent value. The calculation continues in this way until all levels are calculated.

Figure 180 shows the Year dimension from the Sample Basic database. The calculation order is shown on the left. This example assumes that the parent members are not tagged as Dynamic Calc. For a comprehensive discussion of dynamic calculation, see Chapter 25, "Dynamically Calculating Data Values."

Figure 180: Year Dimension from the Sample Basic Database



Jan is the first member in the first branch. Jan has no formula so it is not calculated. The same applies to Feb and Mar, the other two members in the branch.

Analytic Services calculates Qtr1 by consolidating Jan, Feb, and Mar. In this example, these members are added.

Analytic Services then calculates the Qtr2 through Qtr4 branches in the same way.

Finally, Analytic Services calculates the Year member by consolidating the values of Qtr1 through Qtr4. Again, in this example, these members are added.

Determining Member Consolidation

You can choose how Analytic Services consolidates members by applying any calculation operator (+, -, /, *, %, ~) to the members in the database outline.

If an accounts member has a time balance tag (First, Last, or Average), Analytic Services consolidates it accordingly. For information on time balance calculations, see "Calculating First, Last, and Average Values" on page 567.

If a parent member has a label only tag, Analytic Services does not calculate the parent from its children. If a member has a ~ tag, Analytic Services does not consolidate the member up to its parent.

Note: If you use dynamic calculations, Analytic Services may use a different calculation order. For information on the calculation order for dynamically calculated values, see "Calculation Order for Dynamic Calculation" on page 553.

Ordering Dimensions in the Database Outline

To ensure the required calculation results, consider the calculation order of the dimensions in the database outline if you do either of these tasks:

- Use calculation operators to divide (/), multiply (*), or calculate percentages (%) for members in the database outline.
- Place formulas on members in the database outline.

You do not need to consider calculation order if you use only calculation operators to add (+) and subtract (–) members in the database outline and you do not use formulas in the outline.

Placing Formulas on Members in the Database Outline

If you place formulas on members in the database outline, consider the calculation order of the dimensions. A formula that is attached to a member on one dimension may be overwritten by a subsequent calculation on another dimension.

For example, the Sample Basic database has a Measures dimension, tagged as accounts, and a Year dimension, tagged as time. Measures is calculated first, and Year second. If you attach a formula to Margin on the Measures dimension, Analytic Services calculates the formula when it calculates the Measures dimension. Analytic Services then overwrites the formula when it consolidates the Year dimension. For detailed examples of cell calculation order, see "Cell Calculation Order" on page 527.

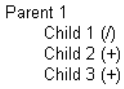
Using the Calculation Operators *, /, and %

If you use calculation operators to multiply (*), divide (/), and calculate percentages (%) for members in the database outline, consider the calculation order of the dimensions. The required calculated values may be overwritten by a subsequent calculation on another dimension.

For example, the Sample Basic database has a Measures dimension, tagged as accounts, and a Year dimension, tagged as time. Measures is calculated first, and Year second. If you multiply members on the Measures dimension, the calculated results may be overwritten when Analytic Services consolidates values on the Year dimension. For detailed examples of cell calculation order, see "Cell Calculation Order" on page 527.

When you use a multiplication (*), division (/), or percentage (%) operator to consolidate members, carefully order the members in the branch to achieve the required result.

Figure 181: Calculation Operators in the Database Outline



In the above example, assume that the user wants to divide the total of Child 2 and Child 3 by Child 1. However, if Child 1 is the first member, Analytic Services starts with Child 1, taking the value of Parent 1 (currently #MISSING) and dividing it by Child 1. The result is #MISSING. Analytic Services then adds Child 2 and Child 3. Obviously, this result is not the required one.

To calculate the correct result, make Child 1 the last member in the branch. For more information on #MISSING values, see "Consolidating #MISSING Values" on page 1217.

You can apply a formula to a member on the database outline to achieve the same result. However, it is far more efficient to use these calculation operators on members as in the above example.

Avoiding Forward Calculation References

To obtain the calculation results you expect, ensure that the outline does not contain forward calculation references. *Forward calculation references* occur when the value of a calculating member is dependent on a member that Analytic Services has not yet calculated. In these cases, Analytic Services may not produce the required calculation results.

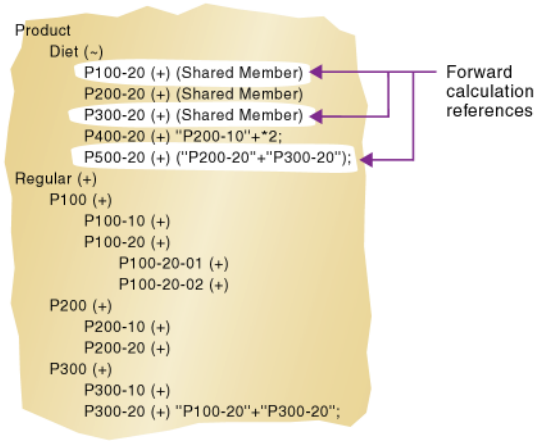
For example, consider this Product dimension:

Figure 182: Example Product Dimension

```
Product
    Diet (~)
        P100-20 (+) (Shared Member)
        P200-20 (+) (Shared Member)
        P300-20 (+) (Shared Member)
        P400-20 (+) "P200-10"*2;
        P500-20 (+) ("P200-20"+"P300-20");
    Regular (+)
        P100 (+)
            P100-10 (+)
            P100-20 (+)
                P100-20-01 (+)
                P100-20-02 (+)
        P200 (+)
            P200-10 (+)
            P200-20 (+)
        P300 (+)
            P300-10 (+)
            P300-20 (+) "P100-20"+"P300-20";
```

This Product dimension has three forward calculation references. Two shared members and one non-shared member have forward calculation references:

Figure 183: Example Product Dimension Showing Forward Calculation References



In Outline Editor, when you verify the outline, Analytic Services identifies shared members with forward calculation references. Verifying the outline does *not* identify non-shared members that have forward calculation references. You can save and use an outline containing forward calculation references.

➤ To verify the outline, see “Verifying Outlines” in the *Essbase Administration Services Online Help*.

Consider the five members under Diet. The members P100-20, P300-20, and P500-20 have forward calculation references:

- P100-20 (+) (Shared Member): Analytic Services calculates the shared member P100-20 before it calculates the real member P100-20. Because the real member P100-20 has children, Analytic Services needs to calculate the real member by adding its children before it can accurately calculate the shared member P100-20.
- P300-20 (+) (Shared Member): Analytic Services calculates the shared member P300-20 before it calculates the real member P300-20. Because the real member P300-20 has a formula, Analytic Services needs to calculate the real member before it can accurately calculate the shared member P300-20.
- P500-20 (+) (“P200-20” + “P300-20”): The formula applied to P500-20 references members that Analytic Services has not yet calculated. One referenced member, P300-20, has its own formula, and Analytic Services needs to calculate P300-20 before it can accurately calculate P500-20. The members P200-20 and P400-20 calculate correctly, as they do not have forward calculation references.
- P200-20 (+) (Shared Member): P200-20 is *not* a forward calculation reference, even though Analytic Services calculates the shared member P200-20 before it calculates the real member P200-20. The real member P200-20 has no calculation dependencies (no children and no formula). Therefore Analytic Services does not need to calculate the real member before the shared member. Analytic Services simply takes the value of the real member.
- P400-20 (+) (“P200-10” * 2): P400-20 is *not* a forward calculation reference, even though the formula that is applied to P400-20 references a member that Analytic Services has not yet calculated. The member referenced in the formula does not itself have calculation dependencies. P200-10 is the only member in the formula, and P200-10 does not itself have children or a formula. Analytic Services accurately calculates P400-20.

To get accurate calculation results for P100-20, P300-20, and P500-20, change the order of members in the outline. By placing the Diet shared members after the Regular members, you ensure that Analytic Services calculates the members in the required order.

Figure 184: Changed Product Dimension Without Forward Calculation References

```
Product
    Regular (+)
        P100 (+)
            P100-10 (+)
            P100-20 (+)
                P100-20-01 (+)
                P100-20-02 (+)
        P200 (+)
            P200-10 (+)
            P200-20 (+)
        P300 (+)
            P300-10 (+)
            P300-20 (+) "P100-20"+"P300-20"
    Diet (~)
        P100-20 (+) (Shared Member)
        P200-20 (+) (Shared Member)
        P300-20 (+) (Shared Member)
        P400-20 (+) "P200-10"*2;
        P500-20 (+) ("P200-20"+"P300-20");
```

Now Analytic Services calculates as follows:

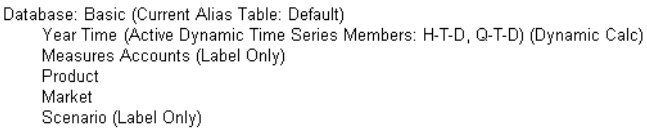
- The real member P100-20 before it calculates the shared member P100-20. So, P100-20 no longer has a forward calculation reference.
- The real member P300-20 before the shared member P300-20. So, P300-20 no longer has a forward calculation reference.
- The referenced member with a formula, P300-20, before the member P500-20. So, P500-20 no longer has a forward calculation reference.

Block Calculation Order

Analytic Services calculates blocks in the order in which the blocks are numbered. Analytic Services takes the first sparse dimension in a database outline as a starting point. It defines the sparse member combinations from this first dimension.

In the Sample Basic database, Product is the first sparse dimension in the database outline.

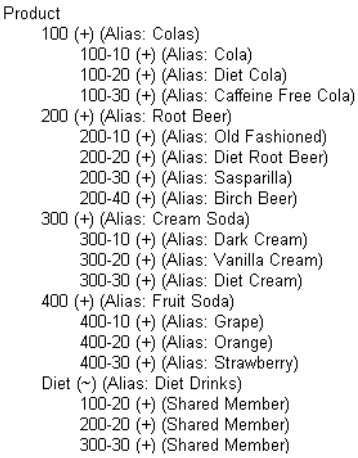
Figure 185: Dimensions in the Sample Basic Database



Note: The attribute dimensions in the Sample Basic outline (not shown in the figure above), are not included in the database consolidation and do not affect block calculation order. For a comprehensive discussion of attribute dimensions, see Chapter 10, "Working with Attributes".

Product has 19 members (excluding the shared members, for which Analytic Services does not create data blocks). Therefore, the first 19 data blocks in the database are numbered according to the calculation order of members in the Product dimension.

Figure 186: Product Dimension from the Sample Basic Database



The other sparse dimension is Market. The first 19 data blocks contain the first member to be calculated in the Market dimension, which is New York.

This table shows the sparse member combinations for the first 5 of these 19 data blocks.

Block #	Product Member	Market Member
0	Cola (100-10)	New York
1	Diet Cola (100-20)	New York
2	Caffeine Free Cola (100-30)	New York
3	Colas (100)	New York
4	Old Fashioned (200-10)	New York

The next member in the Market dimension is Massachusetts. Analytic Services creates the next 19 data blocks for sparse combinations of each Product member and Massachusetts.

This table shows the sparse member combinations for the block numbers 19 through 23.

Block #	Product Member	Market Member
19	Cola (100-10)	Massachusetts
20	Diet Cola (100-20)	Massachusetts
21	Caffeine Free Cola (100-30)	Massachusetts
22	Colas (100)	Massachusetts
23	Old Fashioned (200-10)	Massachusetts

Analytic Services continues until blocks have been created for all combinations of sparse dimension members for which at least one data value exists.

Analytic Services creates a data block only if at least one value exists for the block. For example, if no data values exist for Old Fashioned Root Beer (200-10) in Massachusetts, then Analytic Services does not create a data block for 200-10 -> Massachusetts. However, Analytic Services does reserve the appropriate block number for 200-10 -> Massachusetts in case data is loaded for that member combination in the future.

When you run a default calculation (CALC ALL) on a database, each block is processed in order, according to its block number. If you have Intelligent Calculation turned on and if the block does not need to be calculated, then Analytic Services skips the block and moves on to the next block. For a comprehensive discussion of how intelligent calculation is used to optimize performance, see Chapter 55, "Optimizing with Intelligent Calculation".

Data Block Renumbering

Analytic Services renumbers the data blocks when you make any of these changes:

- Move a sparse dimension
- Add a sparse dimension
- Change a dense dimension to a sparse dimension
- Move any member in a sparse dimension
- Delete any member in a sparse dimension
- Add a member to a sparse dimension

Cell Calculation Order

Each data block contains all the dense dimension member values for its unique combination of sparse dimension members. Each data value is contained in a cell of the data block.

The order in which Analytic Services calculates the cells within each block depends on how you have configured the database. How you have configured the database defines the member calculation order of dense dimension members *within each block*. It also defines the calculation order of blocks that represent sparse dimension members.

Use these sections to understand cell calculation order in more detail:

- "Cell Calculation Order: Example 1" on page 528
- "Cell Calculation Order: Example 2" on page 529
- "Cell Calculation Order: Example 3" on page 531
- "Cell Calculation Order: Example 4" on page 533
- "Cell Calculation Order for Formulas on a Dense Dimension" on page 535

Cell Calculation Order: Example 1

Consider the simplest case in which both of these conditions are true:

- No dimensions have time or accounts tags.
- The setting for consolidating #MISSING values is turned on. For an explanation of how and why #MISSING values are consolidated, see "Consolidating #MISSING Values" on page 1217.

Market and Year are both dense dimensions. The table shows a subset of the cells in a data block. Data values have been loaded into the input cells. Analytic Services calculates the shaded cells. The numbers in bold show the calculation order for these cells. The cell with multiple consolidation paths is darkly shaded.

Year -> Market	New York	Massachusetts	East
Jan	112345.00	68754.00	**3**
Feb	135788.00	75643.00	**4**
Mar	112234.00	93456.00	**5**
Qtr1	**1**	**2**	**6**

As described in "Member Calculation Order" on page 517, Analytic Services calculates dense dimensions in the order that they display in the database outline. Assuming that the Year dimension is displayed before the Market dimension in the database outline, the Year dimension is calculated before the Market dimension.

The cells are calculated in this order:

1. Qtr1 -> New York
2. Qtr1 -> Massachusetts
3. Jan -> East
4. Feb -> East
5. Mar -> East
6. Qtr1 -> East

Qtr1 -> East has multiple consolidation paths. It can be consolidation:missing values:effects on calculation order;consolidated on Market or on Year. When consolidated on Market, it is an consolidation of Qtr1 -> New York and Qtr1 -> Massachusetts. When consolidated on Year, it is an consolidation of Jan -> East, Feb -> East, and Mar -> East.

Analytic Services knows that Qtr1 -> East has multiple consolidation paths. Therefore, it calculates Qtr1 -> East only once and uses the consolidation path of the dimension calculated last. In the above example, this dimension is Market.

The results are shown in this table:

Year/Market	New York	Massachusetts	East
Jan	112345.00	68754.00	181099.00
Feb	135788.00	75643.00	211431.00
Mar	112234.00	93456.00	205690.00
Qtr1	360367.00	237853.00	598220.00

Note: Qtr1 -> East has been calculated only once by consolidating the values for Qtr1.

From the calculation order, you can see that if you place a member formula on Qtr1 in the database outline, Analytic Services ignores it when calculating Qtr1 -> East. If you place a member formula on East in the database outline, the formula is calculated when Analytic Services consolidates Qtr1 -> East on the Market consolidation path. If required, you can use a calculation script to calculate the dimensions in the order you choose. For a comprehensive discussion of how to develop and use calculation scripts, see Chapter 27, “Developing Calculation Scripts.”

Cell Calculation Order: Example 2

Consider a second case in which both of these conditions are true:

- No dimensions have time or accounts tags.
- The setting for consolidating #MISSING values is turned off (the default). For more information, see “Consolidating #MISSING Values” on page 1217.

Market and Year are both dense dimensions. The table shows a subset of the cells in a data block. Data values have been loaded into the input cells. Analytic Services calculates the shaded cells. The numbers in bold show the calculation order for these cells. The cell with multiple consolidation paths is darkly shaded.

Year -> Market	New York	Massachusetts	East
Jan	112345.00	68754.00	**4**
Feb	135788.00	75643.00	**5**
Mar	112234.00	93456.00	**6**
Qtr1	**1**	**2**	**3/7**

As described in "Member Calculation Order" on page 517, Analytic Services calculates dense dimensions in the order they are defined in the database outline. Assuming the Year dimension is positioned before the Market dimension in the database outline, the Year dimension is calculated before the Market dimension.

The cells are calculated in this order:

1. Qtr1 -> New York
2. Qtr1 -> Massachusetts
3. Qtr1 -> East
4. Jan -> East
5. Feb -> East
6. Mar -> East
7. Qtr1 -> East

In this case Qtr1 -> East is calculated on both the Year and Market consolidation paths. First, it is calculated as an consolidation of Qtr1 -> New York and Qtr1 -> Massachusetts. Second, it is calculated as an consolidation of Jan -> East, Feb -> East, and Mar -> East.

The results are identical to the previous case. However, Qtr1 -> East has been calculated twice. This fact is significant when you need to load data at parent levels. For an example in which data is loaded at the parent level, see "Cell Calculation Order: Example 3" on page 531.

Year/Market	New York	Massachusetts	East
Jan	112345.00	68754.00	**181099.00**
Feb	135788.00	75643.00	**211431.00**
Mar	112234.00	93456.00	**205690.00**
Qtr1	**360367.00**	**237853.00**	**598220.00**

From the calculation order, you can see that if you place a member formula on Qtr1 in the database outline, its result is overwritten when Analytic Services consolidates Qtr1 -> East on the Market consolidation path. If you place a member formula on East in the database outline, the result is retained because the Market consolidation path is calculated last.

Cell Calculation Order: Example 3

Consider the previous case in which both of these conditions are true:

- No dimensions have time or accounts tags.
- The setting for consolidating #MISSING values is turned off (the default). For an explanation of how and why #MISSING values are consolidated, see "Consolidating #MISSING Values" on page 1217.
- Data values have been loaded at a parent levels.

Market and Year are both dense dimensions. The table shows a subset of the cells in a data block. Data values have been loaded into cells at the parent level.

Year -> Market	New York	Massachusetts	East
Jan	#MISSING	#MISSING	181099.00
Feb	#MISSING	#MISSING	211431.00
Mar	#MISSING	#MISSING	205690.00
Qtr1	#MISSING	#MISSING	

As described in "Member Calculation Order" on page 517, Analytic Services calculates dense dimensions in the order that they are defined in the database outline. Assuming the Year dimension is positioned before the Market dimension in the database outline, the Year dimension is calculated before the Market dimension.

The cells are calculated in the same order as in Example 2. Qtr1 -> East is calculated on both the Year and Market consolidation paths.

Because the setting for consolidating #MISSING values is turned off, Analytic Services does not consolidate the #MISSING values. Thus, the data that is loaded at parent levels is not overwritten by the #MISSING values below it.

However, if any of the child data values were not #MISSING, these values are consolidated and overwrite the parent values. For example, if Jan -> New York contains 50000.00, this value overwrites the values loaded at parent levels.

Analytic Services first correctly calculates the Qtr1 -> East cell by consolidating Jan -> East, Feb -> East, and Mar -> East. Second, it calculates on the Market consolidation path. However, it does not consolidate the #MISSING values in Qtr1 -> New York and Qtr1 -> Massachusetts and so the value in Qtr1 -> East is not overwritten.

This table shows the results:

Year/Market	New York	Massachusetts	East
Jan	#MISSING	#MISSING	181099.00
Feb	#MISSING	#MISSING	211431.00
Mar	#MISSING	#MISSING	205690.00
Qtr1	#MISSING	#MISSING	**598220.00**

Analytic Services needs to calculate the Qtr1 -> East cell twice in order to ensure that a value is calculated for the cell. If Qtr1 -> East is calculated according to only the last consolidation path, the result is #MISSING, which is not the required result.

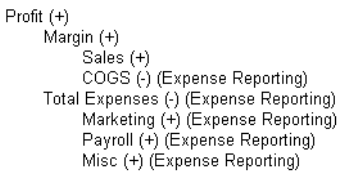
Cell Calculation Order: Example 4

Consider a case in which all of these conditions are true:

- The Year dimension is tagged as time.
- The Measures dimension is tagged as accounts.
- The setting for consolidating #MISSING values is turned off (the default). For an example of how and why #MISSING values are consolidated, see "Consolidating #MISSING Values" on page 1217.

Figure 187 shows the Profit branch of the Measures dimension in the Sample Basic database. This example assumes that Total Expenses is not a Dynamic Calc member. For more information on Dynamic Calc members, see Chapter 25, "Dynamically Calculating Data Values."

Figure 187: Profit Branch of the Measures Dimension in the Sample Basic Database



This table shows a subset of the cells in a data block. Data values have been loaded into the input cells. Analytic Services calculates the shaded cells. The numbers in bold show the calculation order for these cells. Cells with multiple consolidation paths are darkly shaded.

The Marketing, Payroll, and Misc Expenses values have been loaded at the Qtr1, parent level.

Measures/Year	Jan	Feb	Mar	Qtr1
Sales	31538	32069	32213	**13**
COGS	14160	14307	14410	**14**
Margin	**1**	**4**	**7**	**10/15**
Marketing	#MISSING	#MISSING	#MISSING	15839
Payroll	#MISSING	#MISSING	#MISSING	12168
Misc	#MISSING	#MISSING	#MISSING	233
Total Expenses	**2**	**5**	**8**	**11/16**
Profit	**3**	**6**	**9**	**12/17**

As described in "Member Calculation Order" on page 517, Analytic Services calculates a dimension tagged as accounts first, followed by a dimension tagged as time. Therefore, in the above example, Measures is calculated before Year.

Three cells have multiple consolidation paths:

- Margin -> Qtr1
- Total Expenses -> Qtr1
- Profit -> Qtr1

Because the setting for consolidating #MISSING values is turned off, Analytic Services does not consolidate the #MISSING values. Thus, any data that is loaded at parent levels is not overwritten by the #MISSING values and Analytic Services calculates the three cells with multiple consolidation paths twice.

The results are shown in this table.

Measures -> Year	Jan	Feb	Mar	Qtr1
Sales	31538	32069	32213	**95820**
COGS	14160	14307	14410	**42877**
Margin	**17378**	**17762**	**17803**	**52943**
Marketing	#MISSING	#MISSING	#MISSING	15839
Payroll	#MISSING	#MISSING	#MISSING	12168
Misc	#MISSING	#MISSING	#MISSING	233
Total Expenses				**28240**
Profit	**17378**	**17762**	**17803**	**52943**

From the calculation order, you can see that if you place a member formula on, for example, Margin in the database outline, its result is overwritten by the consolidation on Qtr1.

Cell Calculation Order for Formulas on a Dense Dimension

The cell calculation order within a data block is not affected by formulas on members. When Analytic Services encounters a formula in a data block, it locks any other required data blocks, calculates the formula, and proceeds with the data block calculation.

When placing a formula on a dense dimension member, carefully consider the cell calculation order. As described in the examples above, the dimension calculated last overwrites previous cell calculations for cells with multiple consolidation paths. If required, you can use a calculation script to change the order in which the dimensions are calculated. For a comprehensive discussion of how to develop and use calculation scripts, see Chapter 27, “Developing Calculation Scripts.”

For a comprehensive discussion of how to develop and use formulas, see Chapter 22, “Developing Formulas.”

Calculation Passes

Whenever possible, Analytic Services calculates a database in one calculation pass through the database. Thus, it reads each of the required data blocks into memory only once, performing all relevant calculations on the data block and saving it. However, in some situations, Analytic Services needs to perform more than one calculation pass through a database. On subsequent calculation passes, Analytic Services brings data blocks back into memory, performs further calculations on them, and saves them again.

When you perform a default, full calculation of a database (CALC ALL), Analytic Services attempts to calculate the database in one calculation pass. If you have dimensions that are tagged as accounts or time, Analytic Services may have to do more than one calculation pass through the database.

This table shows the number of calculation passes Analytic Services performs if you have dimensions that are tagged as time or accounts, and you have at least one formula on the accounts dimension.

Dimension Tagged As:		Calculation Passes	During each calculation pass, Analytic Services calculates based on:
Accounts	**Time**		
Dense or Sparse	None	1	All dimensions
Dense	Dense	1	All dimensions
Dense	Sparse	2	Pass 1: Accounts and time dimensions Pass 2: Other dimensions
Sparse	Sparse	2	Pass 1: Accounts and time dimensions Pass 2: Other dimensions
Sparse	Dense	2	Pass 1: Accounts dimension Pass 2: Other dimensions

If you are using formulas that are tagged as Two-Pass, Analytic Services may need to do an *extra* calculation pass to calculate these formulas. For a comprehensive discussion of two-pass calculations, see "Using Two-Pass Calculation" on page 1205.

When you use a calculation script to calculate a database, the number of calculation passes Analytic Services needs to perform depends upon the calculation script. For a discussion of single pass and multiple pass calculations, see "Calculation Passes" on page 536 and "Understanding Multiple-Pass Calculations" on page 1234. For information on grouping formulas and calculations, see "Grouping Formulas and Calculations" on page 593.

When you calculate a database, Analytic Services automatically displays the calculation order of the dimensions for each pass through the database and tells you how many times Analytic Services has cycled through the database during the calculation. Analytic Services displays this information in the ESSCMD window and in the application log.

➤ To display the application log, see "Viewing the Analytic Server and Application Logs" on page 997.

For each data block, Analytic Services decides whether to do a dense or a sparse calculation. The type of calculation it chooses depends on the type of values within the data block. When you run a default calculation (CALC ALL) on a database, each block is processed in order, according to its block number.

Analytic Services calculates the blocks using this procedure:

- If you have Intelligent Calculation turned on and if the block does not need to be calculated (if it is marked as *clean*), then Analytic Services skips the block and moves on to the next block. For a comprehensive discussion of intelligent calculation, see Chapter 55, "Optimizing with Intelligent Calculation".
- If the block needs recalculating, Analytic Services checks to see if the block is a level 0, an input, or an upper level block. For definitions of level 0, input, and upper level blocks, see "Data Storage in Data Blocks" on page 516.
- If the block is a level 0 block or an input block, Analytic Services performs a dense calculation on the block. Each cell in the block is calculated. For detailed examples of how Analytic Services calculates cells, see "Cell Calculation Order" on page 527.

- If the block is an upper level block, Analytic Services either consolidates the values or performs a sparse calculation on the data block.

  The sparse member combination of each upper level block contains at least one parent member. Analytic Services consolidates or calculates the block based on the parent member's dimension. For example, if the upper level block is for Product -> Florida from the Sample Basic database, then Analytic Services chooses the Product dimension.

  If the sparse member combination for the block has more than one parent member, Analytic Services chooses the last dimension in the calculation order that includes a parent member. For example, if the block is for Product -> East and you perform a default calculation on the Sample Basic database, Analytic Services chooses the Market dimension, which contains East. The Market dimension is last in the default calculation order because it is placed after the Product dimension in the database outline. For an explanation of the order in which Analytic Services calculates members, see "Member Calculation Order" on page 517.

  Based on the chosen sparse dimension, Analytic Services either consolidates the values or performs a sparse calculation on the data block:

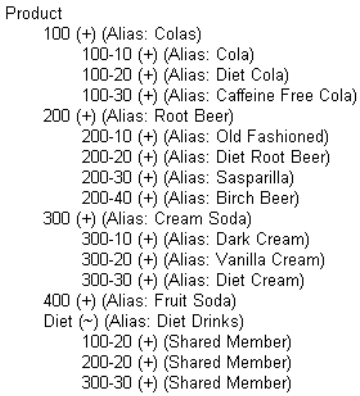
  – If a formula is applied to the data block member on the chosen sparse dimension, Analytic Services performs a formula calculation on the sparse dimension. Analytic Services evaluates each cell in the data block. The formula affects only the member on the sparse dimension, so overall calculation performance is not significantly affected.

  – If the chosen sparse dimension is a default consolidation, Analytic Services consolidates the values, taking the values of the previously calculated child data blocks.

Calculation of Shared Members

Shared members are those that share data values with other members. For example, in the Sample Basic database, Diet Cola, Diet Root Beer, and Diet Cream are consolidated under two different parents. They are consolidated under Diet. They are also consolidated under their individual product types—Colas, Root Beer, and Cream Soda.

Figure 188: Calculating Shared Members



The members under the Diet parent are shared members. For a comprehensive discussion of shared members, see “Understanding Shared Members” on page 164.

A calculation on a shared member is a calculation on the real member. If you use the FIX command to calculate a subset of a database and the subset includes a shared member, Analytic Services calculates the real member.

Chapter 25

Dynamically Calculating Data Values

This chapter explains how you calculate data values dynamically in block storage databases and how you benefit from doing so. Dynamically calculating some of the values in a database can significantly improve the performance of an overall database calculation. This chapter is not relevant to aggregate storage databases.

The information in this chapter assumes that you are familiar with the concepts of member combinations, dense and sparse dimensions, and data blocks. For a comprehensive discussion of these concepts, see Chapter 2, "Understanding Multidimensional Databases."

This chapter includes the following sections:

Understanding Dynamic Calculation

When you design the overall database calculation, it may be more efficient to calculate some member combinations when you retrieve their data, instead of pre-calculating the member combinations during a batch database calculation.

In Analytic Services, you can define a member to have a *dynamic calculation*. This definition tells Analytic Services to calculate a data value for the member as users request it. Dynamic calculation shortens batch database calculation time, but may increase retrieval time for the dynamically calculated data values. See "Reducing the Impact on Retrieval Time" on page 558 for information on how to analyze and manage the effect of dynamic calculation.

In Analytic Services you specify dynamic calculations on a per-member basis. You can define a member in the database outline as one of two types of a dynamically calculated member:

- Dynamic Calc
- Dynamic Calc and Store

Use the rest of this section to gain a basic understanding of dynamic calculation:

Understanding Dynamic Calc Members

For a member that is tagged as Dynamic Calc, Analytic Services does not calculate its data value during a batch database calculation; for example, during a CALC ALL. Instead, Analytic Services calculates the data value upon retrieval; for example, when you retrieve the data into Spreadsheet Add-in or Spreadsheet Services.

Specifically, Analytic Services calculates a data value dynamically when you request the data value in either of two ways:

- By retrieving the data value into Spreadsheet Add-in or Spreadsheet Services
- By running a report script that displays the data value

Analytic Services does not store the calculated value; it recalculates the value for each subsequent retrieval.

Understanding Dynamic Calc and Store Members

Analytic Services calculates the data value for a member that is tagged as Dynamic Calc and Store when you retrieve the data, in the same way as for a Dynamic Calc member. For a Dynamic Calc and Store member, however, Analytic Services stores the data value that is calculated dynamically. Subsequent retrievals of that data value do not require recalculation, unless Analytic Services detects that the value needs recalculating.

Recalculation of Data

When Analytic Services detects that the data value for a Dynamic Calc and Store member needs recalculating, it places an indicator on the data block that contains the value, so that Analytic Services knows to recalculate the block on the next retrieval of the data value.

Analytic Services places the indicator on the data block containing the value and not on the data value itself. In other words, Analytic Services tracks Dynamic Calc and Store members at the data block level. For detailed information on data blocks, see "Data Blocks and the Index System" on page 64.

If the data block needs recalculating, Analytic Services detects the need and places an indicator on the data block when any of the following occur:

- You perform a batch calculation.
- You restructure the database.
- You use the CLEARBLOCK DYNAMIC calculation command. For more information on the CLEARBLOCK command, see the *Technical Reference*.

Analytic Services recalculates the indicated data blocks when you next retrieve the data value.

Effect of Updated Values on Recalculation

Analytic Services does *not* detect that a data block needs recalculating and does *not* place an indicator on the data block when you update the data; that is, updated blocks are recalculated only during the next batch calculation. Consider the following two scenarios:

- You do a data load.
- You do a Lock and Send from Spreadsheet Services.

If you load data into the children of a Dynamic Calc and Store member, and the member is a consolidation of its child members, Analytic Services does *not* know to recalculate the Dynamic Calc and Store member during the next retrieval. The parent member is recalculated only during the next batch calculation.

After loading data, you need to perform a batch calculation of the database or use the CLEARBLOCK DYNAMIC calculation command to ensure that the Dynamic Calc and Store members are recalculated. For more information on the CLEARBLOCK command, see the *Technical Reference*.

Retrieving the Parent Value of Dynamically Calculated Child Values

If you retrieve a parent value that is calculated from Dynamic Calc or Dynamic Calc and Store child members, Analytic Services must dynamically calculate the child member combinations before calculating the parent value. Analytic Services does not store the child values, even if they are Dynamic Calc and Store members.

For example, assume that Market is a parent member and that East and West are Dynamic Calc and Store child members that consolidate up to Market. When you retrieve a data value for Market, Analytic Services calculates East and West, even though you have not specifically retrieved them. However, Analytic Services does not store the values of East or West.

Benefitting from Dynamic Calculation

Dynamically calculating some database values can significantly improve the performance of an overall database calculation.

Calculating some data values dynamically, achieves the following advantages:

- It reduces the batch calculation time of the database because Analytic Services has fewer member combinations to calculate.
- It reduces disk usage because Analytic Services stores fewer calculated data values. Database size and index size are reduced.
- It reduces database restructure time. For example, adding or deleting a Dynamic Calc member in a dense dimension does not change the data block size, and so Analytic Services does not need to restructure the database for such additions and deletions. For an explanation of the relationship between the use of Dynamic Calc members and database restructures, see "Restructuring Databases" on page 564.
- It reduces the time that is required to back up the database. Because database size is reduced, Analytic Services takes less time to perform a backup.

Data values that Analytic Services calculates dynamically can take longer to retrieve. You can estimate the retrieval time for dynamically calculated members. For information on how to analyze and manage the effect of dynamic calculation, see "Reducing the Impact on Retrieval Time" on page 558.

Using Dynamic Calculation

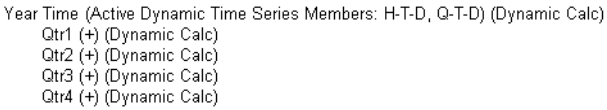
You can tag any member as Dynamic Calc or Dynamic Calc and Store, except the following:

- Level 0 members that do not have a formula
- Label-Only members
- Shared members

Which members you choose to calculate dynamically depends on the database structure and on the balance between (1) the need for reduced calculation time and disk usage and (2) the need for speedy data retrieval for users. For guidelines on choosing which members to calculate dynamically, see "Choosing Values to Calculate Dynamically" on page 546.

Outline Editor shows which members are Dynamic Calc and which members are Dynamic Calc and Store.

Figure 189: Sample Basic Outline Showing Dynamic Calc Members



In Spreadsheet Add-in or Spreadsheet Services, users can display visual cues to distinguish dynamically calculated values. For more information, see the Spreadsheet online help.

When developing spreadsheets that include dynamically calculated values, spreadsheet designers may want to use the spreadsheet Navigate Without Data option, so that Analytic Services does not dynamically calculate and store values while test spreadsheets are being built.

Choosing Values to Calculate Dynamically

Dynamically calculating some data values decreases calculation time, lowers disk usage, and reduces database restructure time, but increases retrieval time for dynamically calculated data values.

Use the guidelines described in the following sections when deciding which members to calculate dynamically:

Dense Members and Dynamic Calculation

Consider making the following changes to members of dense dimensions:

- Tag upper level members of dense dimensions as Dynamic Calc.
- Try tagging level 0 members of dense dimensions with simple formulas as Dynamic Calc, and assess the increase in retrieval time.
- Do *not* tag members of dense dimensions as Dynamic Calc and Store.

Simple formulas are formulas that do not require Analytic Services to perform an expensive calculation. Formulas containing, for example, financial functions or cross-dimensional operators (->) are complex formulas.

Sparse Members and Dynamic Calculation

Consider making the following changes to members of sparse dimensions:

- Tag some upper level members of sparse dimensions that have six or fewer children as Dynamic Calc or Dynamic Calc and Store.
- Tag sparse-dimension members with complex formulas as Dynamic Calc or Dynamic Calc and Store. A complex formula requires Analytic Services to perform an expensive calculation. For example, any formula that contains a financial function is a complex formula. For a discussion of complex formulas (definition, guidelines for use, and effect on performance), see "Using Complex Formulas" on page 1195.
- Tag upper level members in a dimension that you frequently restructure as Dynamic Calc or Dynamic Calc and Store.
- Do *not* tag upper level, sparse-dimension members that have 20 or more descendants as Dynamic Calc or Dynamic Calc and Store.

For recommendations in regard to use of Dynamic Calc and Dynamic Calc And Store, see "Choosing Between Dynamic Calc and Dynamic Calc and Store" on page 550.

Two-Pass Members and Dynamic Calculation

To reduce the amount of time needed to perform batch calculations, tag two-pass members as Dynamic Calc. You can tag any Dynamic Calc or Dynamic Calc and Store member as two-pass, even if the member is not on an accounts dimension.

For a comprehensive discussion of two-pass calculation, see "Using Two-Pass Calculation" on page 1205. For details about the interaction of members tagged as two-pass and attribute members, see "Comparing Attribute and Standard Dimensions" on page 188.

Parent-Child Relationships and Dynamic Calculation

If a parent member has a single child member and you tag the *child* as Dynamic Calc, you must tag the parent as Dynamic Calc. Similarly, if you tag the *child* as Dynamic Calc and Store, you must tag the parent as Dynamic Calc and Store. However, if a parent member has a single child member and the parent is a Dynamic Calc or Dynamic Calc and Store member, you do not have to tag the child as Dynamic Calc or Dynamic Calc and Store.

Calculation Scripts and Dynamic Calculation

When Analytic Services calculates, for example, a CALC ALL or CALC DIM statement in a calculation script, it bypasses the calculation of Dynamic Calc and Dynamic Calc and Store members.

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

If you specify a Dynamic Calc or Dynamic Calc and Store member explicitly in a calculation script, the calculation script fails. You cannot do a calculation script calculation of a Dynamic Calc or Dynamic Calc and Store member. If you want to use a calculation script to calculate a member explicitly, do not tag the member as Dynamic Calc. For example, the following calculation script is valid only if Qtr1 is not a Dynamic Calc member:

```
FIX (East, Colas)
Qtr1;
ENDFIX
```

Formulas and Dynamically Calculated Members

You *can* include a dynamically calculated member in a formula when you apply the formula to the database outline. For example, if Qtr1 is a Dynamic Calc member, you can place the following formula on Qtr1 in the database outline:

```
Qtr1 = Jan + Feb;
```

You *cannot* make a dynamically calculated member the target of a formula calculation in a calculation script; Analytic Services does not reserve memory space for a dynamically calculated value and, therefore, cannot assign a value to it. For example, if Qtr1 is a Dynamic Calc or Dynamic Calc and Store member, Analytic Services displays a syntax error if you include the following formula in a calculation script:

```
Qtr1 = Jan + Feb;
```

However, if Qtr1 is a Dynamic Calc or Dynamic Calc and Store member and Year is neither Dynamic Calc nor Dynamic Calc and Store, you can use the following formula in a calculation script:

```
Year = Qtr1 + Qtr2;
```

This formula is valid because Analytic Services is not assigning a value to the dynamically calculated member.

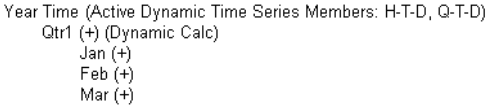
Note: When you reference a dynamically calculated member in a formula in the database outline or in a calculation script, Analytic Services interrupts the regular calculation to do the dynamic calculation. This interruption can significantly lower calculation performance.

Dynamically Calculated Children

If the calculation of a member depends on the calculation of Dynamic Calc or Dynamic Calc and Store child members, Analytic Services has to calculate the child members first during the batch database calculation in order to calculate the parent. Therefore, there is no reduction in regular calculation time. This requirement applies to members of sparse dimensions and members of dense dimensions.

For example, consider the following outline:

Figure 190: Sample Basic Outline, Showing Qtr1 as a Dynamic Calc Member



Qtr1 is a Dynamic Calc member. Its children, Jan, Feb, and Mar, are not dynamic members, and its parent, Year, is not a dynamic member. When Analytic Services calculates Year during a batch database calculation, it must consolidate the values of its children, including Qtr1. Therefore, it must take the additional time to calculate Qtr1 even though Qtr1 is a Dynamic Calc member.

Choosing Between Dynamic Calc and Dynamic Calc and Store

In most cases you can optimize calculation performance and lower disk usage by using Dynamic Calc members instead of Dynamic Calc and Store members. However, in specific situations, using Dynamic Calc and Store members is optimal:

- “Recommendations for Sparse Dimension Members” on page 550
- “Recommendations for Members with Specific Characteristics” on page 551
- “Recommendations for Dense Dimension Members” on page 552
- “Recommendations for Data with Many Concurrent Users” on page 552

Recommendations for Sparse Dimension Members

In most cases, when you want to calculate a sparse dimension member dynamically, tag the member as Dynamic Calc instead of Dynamic Calc and Store. When Analytic Services calculates data values for a member combination that includes a Dynamic Calc member, Analytic Services calculates only the requested values of the relevant data block. These values can be a subset of the data block.

However, when Analytic Services calculates data values for a member combination that includes a Dynamic Calc and Store member, Analytic Services needs to calculate and store the whole data block, even if the requested data values are a subset of the data block. Thus, the calculation takes longer and the initial retrieval time is greater.

Analytic Services stores only the data blocks that contain the requested data values. If Analytic Services needs to calculate any intermediate data blocks in order to calculate the requested data blocks, it does not store the intermediate blocks.

Calculating the intermediate data blocks can significantly increase the initial retrieval time. For example, in the Sample Basic database, Market and Product are the sparse dimensions. Assume that Market and the children of Market are Dynamic Calc and Store members. When a user retrieves the data value for the member combination Market -> Cola -> Jan -> Actual -> Sales, Analytic Services calculates and stores the Market -> Cola data block. In order to calculate and store Market -> Cola, Analytic Services calculates the intermediate data blocks—East -> Cola, West -> Cola, South -> Cola, and Central -> Cola. Analytic Services does not store these intermediate data blocks.

Recommendations for Members with Specific Characteristics

Using Dynamic Calc and Store may slow initial retrieval; however, subsequent retrievals are faster than for Dynamic Calc members. Use Dynamic Calc and Store instead of Dynamic Calc for the following members:

- An upper-level sparse dimension member with children on a remote database. Analytic Services needs to retrieve the value from the remote database, which increases retrieval time. For a detailed explanation of this situation, see “Dynamically Calculating Data in Partitions” on page 565.
- A sparse dimension member with a complex formula. A complex formula requires Analytic Services to perform an expensive calculation. For example, any formula that contains a financial function or a cross-dimensional member is a complex formula.
- If users frequently retrieve an upper level member of a sparse dimension, speedy retrieval time is very important.

For example, in the Sample Basic database, if most users retrieve data at the Market level, you probably want to tag Market as Dynamic Calc and Store and its children as Dynamic Calc.

Figure 191: Sample Basic Outline, Market is Dynamic Calc and Store Member

Market (Dynamic Calc And Store)
East (+) (Dynamic Calc) (UDAs: Major Market)
West (+) (Dynamic Calc)
South (+) (Dynamic Calc) (UDAs: Small Market)
Central (+) (Dynamic Calc) (UDAs: Major Market)

Recommendations for Dense Dimension Members

Use Dynamic Calc members for dense dimension members. Defining members as Dynamic Calc and Store on a dense dimension provides only a small decrease in retrieval time and in batch calculation time. In addition, database size (disk usage) does not decrease significantly because Analytic Services reserves space in the data block for the data values of the member.

Recommendations for Data with Many Concurrent Users

Use Dynamic Calc members for data with concurrent users. If many users are concurrently retrieving Analytic Services data, the initial retrieval time for Dynamic Calc and Store members can be significantly higher than for Dynamic Calc members.

Dynamic Calc and Store member retrieval time increases as the number of concurrent user retrievals increases. However, Dynamic Calc member retrieval time does not increase as concurrent user retrievals increase.

If many users are concurrently accessing data, you may see significantly lower retrieval times if you use Dynamic Calc members instead of Dynamic Calc and Store members.

Understanding How Dynamic Calculation Changes Calculation Order

Using dynamically calculated data values changes the order in which Analytic Services calculates the values and can have implications for the way you administer a database:

Calculation Order for Dynamic Calculation

When Analytic Services dynamically calculates data values, it calculates the data in an order that is different from the batch database calculation order. For detailed information on calculation order when dynamic calculation is not used, see Chapter 24, "Defining Calculation Order."

During batch calculations, Analytic Services calculates the database in this order:

1. Dimension tagged as accounts
2. Dimension tagged as time
3. Other dense dimensions (in the order they appear in the database outline)
4. Other sparse dimensions (in the order they appear in the database outline)
5. Two-pass calculations

For dynamically calculated values, on retrieval, Analytic Services calculates the values by calculating the database in the following order:

1. Sparse dimensions
   - If the dimension tagged as time is sparse and the database outline uses time series data, Analytic Services bases the sparse calculation on the time dimension.
   - Otherwise, Analytic Services bases the calculation on the dimension that it normally uses for a batch calculation.

2. Dense dimensions
   a. Dimension tagged as accounts, if dense
   b. Dimension tagged as time, if dense
   c. Time series calculations
   d. Remaining dense dimensions
   e. Two-pass calculations
   f. Attributes

      Note: If your data retrieval uses attribute members, the last step in the calculation order is the summation of the attributes. However, the use of attribute members in your query causes Analytic Services to disregard the value of the Time Balance member in the dynamic calculations.

      During retrievals that do not use attributes, the value of the Time Balance member is applied to the calculations.The difference in calculation procedure between the use and non-use of attribute members generates different results for any upper level time members that are dynamically calculated.

During retrievals that do not use attributes, these dynamically calculated members are calculated in the last step and, therefore, apply the time balance functionality properly. However, during retrievals that do use attributes the summation of the attribute is the last step applied. The difference in calculation order produces two different, predictable results for upper level time members that are dynamically calculated.

Calculation Order for Dynamically Calculating Two-Pass Members

Consider the following information to ensure that Analytic Services produces the required calculation result when it dynamically calculates data values for members that are tagged as two-pass. For a comprehensive discussion of two-pass calculations, see “Using Two-Pass Calculation” on page 1205.

If more than one Dynamic Calc or Dynamic Calc and Store dense dimension member is tagged as two-pass, Analytic Services performs the dynamic calculation in the first pass, and then calculates the two-pass members in the following order:

1. Two-pass members in the accounts dimension, if any exist.
2. Two-pass members in the time dimension, if any exist.
3. Two-pass members in the remaining dense dimensions in the order in which the dimensions appear in the outline.

For example, in the Sample Basic database, assume the following:

- Margin% in the dense Measures dimension (the dimension tagged as accounts) is tagged as both Dynamic Calc and two-pass.
- Variance in the dense Scenario dimension is tagged as both Dynamic Calc and two-pass.

Analytic Services calculates the accounts dimension member first. So, Analytic Services calculates Margin% (from the Measures dimension) and then calculates Variance (from the Scenario dimension).

If Scenario is a sparse dimension, Analytic Services calculates Variance first, following the regular calculation order for dynamic calculations. For a description of calculation order for dynamic calculations, see "Calculation Order for Dynamic Calculation" on page 553. Analytic Services then calculates Margin%.

This calculation order does not produce the required result because Analytic Services needs to calculate Margin % -> Variance using the formula on Margin %, and not the formula on Variance. You can avoid this problem by making Scenario a dense dimension. This problem does not occur if the Measures dimension (the accounts dimension) is sparse, because Analytic Services still calculates Margin% first.

Calculation Order for Asymmetric Data

Because the calculation order used for dynamic calculations differs from the calculation order used for batch database calculations, in some database outlines you may get different calculation results if you tag certain members as Dynamic Calc or Dynamic Calc and Store. These differences happen when Analytic Services dynamically calculates asymmetric data.

Symmetric data calculations produce the same results no matter which dimension is calculated. Asymmetric data calculations calculate differently along different dimensions.

Table 25: Example of a Symmetric Calculation

Time -> Accounts	Jan	Feb	Mar	Qtr1
Sales	100	200	300	600
COGS	50	100	150	300
Profit (Sales – COGS)	50	100	150	300

The calculation for Qtr1-> Profit produces the same result whether you calculate along the dimension tagged as time or the dimension tagged as accounts. Calculating along the time dimension, add the values for Jan, Feb, and Mar:

```
50+100+150=300
```

Calculating along the accounts dimension, subtract Qtr1 -> COGS from Qtr1 -> Sales:

```
600–300=300
```

Table 26: Example of an Asymmetric Calculation

Market -> Accounts	New York	Florida	Connecticut	East
UnitsSold	10	20	20	50
Price	5	5	5	15
Sales (Price * UnitsSold)	50	100	100	250

The calculation for East -> Sales produces the correct result when you calculate along the Market dimension, but produces an incorrect result when you calculate along the accounts dimension. Calculating along the Market dimension, adding the values for New York, Florida, and Connecticut produces the correct results:

```
50+100+100=250
```

Calculating along the accounts dimension, multiplying the value East -> Price by the value East -> UnitsSold produces incorrect results:

```
15 * 50=750
```

In this outline, East is a sparse dimension, and Accounts is a dense dimension:

```
East
     New York (+)
     Florida (+)
     Connecticut (+)
Accounts
     UnitsSold (~)
     Price (~)
     Sales (~) UnitsSold*Price;
```

If East and Sales are tagged as Dynamic Calc, then Analytic Services calculates a different result than it does if East and Sales are not tagged as Dynamic Calc.

If East and Sales are not Dynamic Calc members, Analytic Services produces the correct result by calculating as follows:

1. The dense Accounts dimension, calculating the values for UnitsSold, Price, and Sales for New York, Florida, and Connecticut.
2. The sparse East dimension, by aggregating the calculated values for UnitsSold, Price, and Sales for New York, Florida, and Connecticut to obtain the Sales values for East.

If East and Sales are Dynamic Calc members, Analytic Services produces an incorrect result by calculating as follows:

1. The sparse East dimension, by aggregating the values for UnitsSold, Price, and Sales for New York, Florida, and Connecticut to obtain the values for East.
2. The values for East -> Sales, by taking the aggregated values in the East data blocks and performing a formula calculation with these values to obtain the value for Sales.

To avoid this problem and ensure that you obtain the required results, do not tag the Sales member as Dynamic Calc or Dynamic Calc and Store.

Reducing the Impact on Retrieval Time

The increase in retrieval time when you dynamically calculate a member of a dense dimension is not significant unless the member contains a complex formula. The increase in retrieval time may be significant when you tag members of sparse dimensions as Dynamic Calc or Dynamic Calc and Store.

The following sections discuss ways you can analyze and manage the effect of Dynamic Calc members on a database:

- "Displaying a Retrieval Factor" on page 558
- "Displaying a Summary of Dynamically Calculated Members" on page 559
- "Increasing Retrieval Buffer Size" on page 559
- "Using Dynamic Calculator Caches" on page 560

Note: For a list of functions that have the most significant effect on query retrieval, see "Choosing Between Member Set Functions and Performance" on page 1216 for details.

Displaying a Retrieval Factor

To help you estimate any increase in retrieval time, Analytic Services calculates a retrieval factor for a database outline when you save the outline. Analytic Services calculates this retrieval factor based on the dynamically calculated data block that is the most expensive for Analytic Services to calculate. The retrieval factor takes into account only aggregations. It does not consider the retrieval impact of formulas.

The retrieval factor is the number of data blocks that Analytic Services must retrieve from disk or from the database in order to calculate the most expensive block. If the database has Dynamic Calc or Dynamic Calc and Store members in dense dimensions only (no Dynamic Calc or Dynamic Calc and Store members in sparse dimensions), the retrieval factor is 1.

An outline with a high retrieval factor (for example, greater than 2000) can cause long delays when users retrieve data. However, the actual impact on retrieval time also depends on how many dynamically calculated data values a user retrieves. The retrieval factor is only an indicator. In some applications, using Dynamic Calc members may reduce retrieval time because the database size and index size are reduced.

Analytic Services displays the retrieval factor value in the application log.

➤ To view an estimated retrieval factor, see "Viewing the Analytic Server and Application Logs" on page 997.

A message similar to this sample indicates a retrieval factor:

```
[Wed Sep 20 20:04:13 2000] Local/Sample///Info (1012710)
Essbase needs to retrieve [1] Essbase kernel blocks in order
to calculate the top dynamically-calculated block.
```

This message tells you that Analytic Services needs to retrieve one block in order to calculate the most expensive dynamically calculated data block.

Displaying a Summary of Dynamically Calculated Members

When you add Dynamic Calc or Dynamic Calc and Store members to a database outline and save the outline, Analytic Services provides a summary of how many members are tagged as Dynamic Calc and Dynamic Calc and Store. Analytic Services displays the summary in the application log.

➤ To view a summary of dynamically calculated members, see "Viewing the Analytic Server and Application Logs" on page 997.

A message similar to this sample is displayed:

```
[Wed Sep 20 20:04:13 2000]Local/Sample///Info(1007125)
The number of Dynamic Calc Non-Store Members = [ 8 6 0 0 2]

[Wed Sep 20 20:04:13 2000]Local/Sample///Info(1007126)
The number of Dynamic Calc Store Members = [ 0 0 0 0 0]
```

This message tells you that there are eight Dynamic Calc members in the first dimension of the database outline, six in the second dimension, and two in the fifth dimension. Dynamic Time Series members are included in this count.

This example does not include Dynamic Calc and Store members.

Increasing Retrieval Buffer Size

When you retrieve data into Spreadsheet Services or use Report Writer to retrieve data, Analytic Services uses the retrieval buffer to optimize the retrieval. Analytic Services processes the data in sections. Increasing the retrieval buffer size can significantly reduce retrieval time because Analytic Services can process larger sections of data at one time.

By default, the retrieval buffer size is 10 KB. However, you may speed up retrieval time if you set the retrieval buffer size greater than 10 KB. For information about sizing the retrieval buffer, see "Setting the Retrieval Buffer Size" on page 1244.

➤ Use any of the following methods to set the retrieval buffer size:

Tool	Topic	Location
Administration Services	Setting the Size of Retrieval Buffers	*Essbase Administration Services Online Help*
MaxL	**alter database**	*Technical Reference*
ESSCMD	SETDBSTATEITEM	*Technical Reference*

Using Dynamic Calculator Caches

By default, when Analytic Services calculates a Dynamic Calc member in a dense dimension (for example, for a query), it writes all blocks needed for the calculation into an area in memory called the dynamic calculator cache. When Analytic Services writes these blocks into the dynamic calculator cache, it expands them to include all Dynamic Calc members in the dense dimensions.

Using the Analytic Services dynamic calculator cache enables centralized control of memory usage for dynamic calculations. Managing data blocks in the dynamic calculator cache also reduces the overall memory space requirement and can improve performance by reducing the number of calls to the operating system to do memory allocations.

Note: The dynamic calculator cache and the calculator cache use different approaches to optimizing calculation performance.

For details about sizing and reviewing dynamic calculator cache usage, see "Sizing the Calculator Cache" on page 1133.

Reviewing Dynamic Calculator Cache Usage

Analytic Services writes two messages to the application log for each data retrieval. As shown in the example in Figure 192, the first message describes the total amount of time required for the retrieval.

Figure 192: Application Log Example of Memory Usage for Data Blocks Containing Dynamic Calc Members

```
[Thu Aug 03 14:33:00 2000]Local/Sample/Basic/aspen/Info(1001065)
Regular Extractor Elapsed Time : [0.531] seconds

[Thu Aug 03 14:33:00 2000]Local/Sample/Basic/aspen/Info(1001401)
Regular Extractor Big Blocks Allocs -- Dyn.Calc.Cache : [30]
non-Dyn.Calc.Cache : [0]
```

If a dynamic calculator cache is used, a second message displays the number of blocks calculated within the data calculator cache (Dyn.Calc.Cache: [*n*]) and the number of blocks calculated in memory outside dynamic calculator cache (non-Dyn.Calc.Cache: [*n*]).

To determine if the dynamic calculator cache is being used effectively, review both of these messages and consider what your settings are in the `essbase.cfg` file. For example, if the message indicates that blocks were calculated outside as well as in a dynamic calculator cache, you may need to increase the DYNCALCCACHEMAXSIZE setting. If the specified maximum size is all that you can afford for all dynamic calculator caches on the server and if using memory outside the calculator cache to complete dynamically calculated retrievals results in unacceptable delays (for example, because of swapping or paging activity), set DYNCALCCACHEWAITFORBLK to TRUE.

You can use the GETPERFSTATS command in ESSCMD to view a summary of dynamic calculator cache activity. See the *Technical Reference* for information about GETPERFSTATS.

Using Dynamic Calculations with Standard Procedures

Using dynamic calculations with standard Analytic Services procedures affects these processes:

- Clearing data and data blocks

  You can use the CLEARBLOCK DYNAMIC command to remove data blocks for Dynamic Calc and Store member combinations.

  You can use the CLEARDATA command to mark Dynamic Calc and Store data blocks, so that Analytic Services knows to recalculate the blocks. The CLEARDATA command has no effect on data values for Dynamic Calc members.

- Copying data

  You cannot copy data to a dynamically calculated data value. You cannot specify a Dynamic Calc or Dynamic Calc and Store member as the target for the DATACOPY calculation command.

- Converting currencies

  You cannot specify a Dynamic Calc or Dynamic Calc and Store member as the target for the CCONV command.

- Loading data

  When you load data, Analytic Services does not load data into member combinations that contain a Dynamic Calc or Dynamic Calc and Store member. Analytic Services skips these members during data load. Analytic Services does not display an error message.

  To place data into Dynamic Calc and Dynamic Calc and Store members, after loading data, you need to ensure that Analytic Services recalculates Dynamic Calc and Store members. For an explanation of how to ensure that Analytic Services recalculates Dynamic Calc and Store members, see "Effect of Updated Values on Recalculation" on page 544.

- Exporting data

  Analytic Services does not calculate dynamically calculated values before exporting data. Analytic Services does not export values for Dynamic Calc members. Analytic Services exports values for Dynamic Calc and Store members only if a calculated value exists in the database from a previous user retrieval of the data.

- Reporting data

  Analytic Services cannot use the SPARSE data extraction method for dynamically calculated members. The SPARSE data extraction method optimizes performance when a high proportion of the reported data rows are #MISSING. For more information, see the <SPARSE command in the *Technical Reference*.

- Including dynamic members in calculation scripts

  When calculating a database, Analytic Services skips the calculation of any Dynamic Calc or Dynamic Calc and Store members. Analytic Services displays an error message if you attempt to do a member calculation of a Dynamic Calc or Dynamic Calc and Store member in a calculation script. For an explanation of the relationship of Dynamic Calc and Dynamic Calc and Store members and calculation passes, see "Calculation Scripts and Dynamic Calculation" on page 548.

Creating Dynamic Calc and Dynamic Calc and Store Members

➤ To create Dynamic Calc and Dynamic Calc and Store members using Outline Editor, see "Setting Member Storage Properties" in *Essbase Administration Services Online Help*.

➤ To create Dynamic Calc and Dynamic Calc and Store members during a dimension build, in the dimension build data file, use the property X for Dynamic Calc and the property V for Dynamic Calc and Store. For information on how to place X and V in the data source, see "Using the Data Source to Set Member Properties" on page 379.

Restructuring Databases

When you add a Dynamic Calc member to a dense dimension, Analytic Services does not reserve space in the data block for the member's values. Therefore, Analytic Services does not need to restructure the database. However, when you add a Dynamic Calc and Store member to a dense dimension, Analytic Services does reserve space in the relevant data blocks for the member's values and therefore needs to restructure the database.

When you add a Dynamic Calc or a Dynamic Calc and Store member to a sparse dimension, Analytic Services updates the index, but does not change the relevant data blocks. For information on managing the database index, see "Index Manager" on page 1025.

Analytic Services can save changes to the database outline significantly faster if it does not have to restructure the database.

In the following cases, Analytic Services does not restructure the database. Analytic Services only has to save the database outline, which is very fast. Analytic Services does *not* restructure the database or change the index when you do any of the following:

- Add, delete, or move a dense dimension Dynamic Calc member. (But Analytic Services *does* restructure the database if the member is Dynamic Calc and Store.)
- Change the storage property of a dense dimension member from Dynamic Calc and Store member to a non-dynamic storage property
- Change the storage property of a sparse dimension Dynamic Calc or Dynamic Calc and Store member to a non-dynamic storage property
- Rename any Dynamic Calc or Dynamic Calc and Store member

In the following cases, Analytic Services does not restructure the database, but does have to restructure the database index. Restructuring the index is significantly faster than restructuring the database.

Analytic Services restructures only the database index when you do either of the following:

- Add, delete, or move sparse dimension Dynamic Calc or Dynamic Calc and Store members
- Change the storage property of a dense dimension member from a non-dynamic value to Dynamic Calc and Store

However, Analytic Services does restructure the database when you do any of the following:

- Add, delete, or move a dense dimension Dynamic Calc and Store member. (But Analytic Services does *not* restructure the database if the member is Dynamic Calc.)
- Change a dense dimension Dynamic Calc and Store member to a Dynamic Calc member
- Change a dense dimension Dynamic Calc member to a Dynamic Calc and Store member
- Change the storage property of a non-dynamic member in a dense dimension to Dynamic Calc
- Change the storage property of a dense dimension from Dynamic Calc member to a non-dynamic value
- Change the storage property of a non-dynamic member in a sparse dimension Dynamic Calc or Dynamic Calc and Store

For detailed information on the types of database restructuring, see "Types of Database Restructuring" on page 1148.

Dynamically Calculating Data in Partitions

You can define Dynamic Calc and Dynamic Calc and Store members in transparent, replicated, or linked regions of the partitions. For a comprehensive discussion of partitions, see Chapter 13, "Designing Partitioned Applications".

For example, you might want to tag an upper level, sparse dimension member with children that are on a remote database (transparent database partition) as Dynamic Calc and Store. Because Analytic Services needs to retrieve the child values from

the other database, retrieval time is increased. You can use Dynamic Calc instead of Dynamic Calc and Store; however, the impact on subsequent retrieval time might be too great.

For example, assume that the local database is the Corporate database, which has transparent partitions to the regional data for East, West, South, and Central. You can tag the parent member Market as Dynamic Calc and Store.

In a transparent partition, the definition on the remote database takes precedence over any definition on the local database. For example, if a member is tagged as Dynamic Calc in the local database but not in the remote database, Analytic Services retrieves the value from the remote database and does not do the local calculation.

If you are using a replicated partition, then you might want to use Dynamic Calc members instead of Dynamic Calc and Store members. When calculating replicated data, Analytic Services does not retrieve the child blocks from the remote database, and therefore the impact on retrieval time is not great.

Note: When Analytic Services replicates data, it checks the time stamp on each source data block and each corresponding target data block. If the source data block is more recent, Analytic Services replicates the data in the data block. However, for dynamically calculated data, data blocks and time stamps do not exist. Therefore Analytic Services always replicates dynamically calculated data.

Chapter 26

Calculating Time Series Data

This chapter explains how to calculate time series data in block storage databases. For example, you can do inventory tracking by calculating the first and last values for a specific time period. You can also calculate period-to-date values.

This chapter pertains to block storage databases. For block storage databases, time series calculations assume that you have Dynamic Time Series members defined in the outline. Dynamic Time Series members are not supported for aggregate storage database outlines. However, you can perform time series calculations and queries on aggregate storage databases using the MDX PeriodsToDate function. For more information, see the MDX section of the *Technical Reference*.

This chapter includes the following sections:

Calculating First, Last, and Average Values

Using time balance and variance reporting tags on the dimension tagged as accounts, you can tell Analytic Services how to perform time balance calculations on accounts data.

Analytic Services usually calculates a parent in the dimension tagged as time by consolidating or calculating the formulas on the parent's children. However, you can use accounts tags, such as time balance and variance reporting tags, to consolidate a different kind of value. For example, if you tag a parent member in the accounts dimension with a time balance property of First, Analytic Services calculates the member by consolidating the value of the member's first child. For example, in the Sample Basic database, the Opening Inventory member in the Measures dimension (the accounts dimension) has a time balance property of First.

This member represents the inventory at the beginning of the time period. If the time period is Qtr1, Opening Inventory represents the inventory available at the beginning of Jan (the first member in the Qtr1 branch).

To use accounts tags, you must have a dimension tagged as accounts and a dimension tagged as time. You use the First, Last, and Average tags (time balance properties) and the Expense tag (variance reporting property) only on members of a dimension tagged as accounts. The dimensions you tag as time and accounts can be either dense or sparse dimensions.

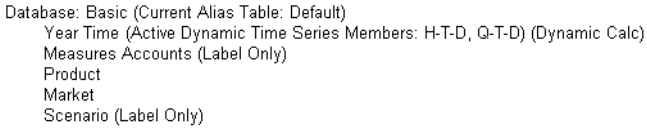
Note: If you are using Intelligent Calculation, changing accounts tags in the database outline does not cause Analytic Services to restructure the database. You may have to tell Analytic Services explicitly to recalculate the required data values. For a discussion of how and why to perform a recalculation, see "Changing Formulas and Accounts Properties" on page 1240.

Specifying Accounts and Time Dimensions

When you tag a dimension as accounts, Analytic Services knows that the dimension contains members with accounts tags. When you tag a dimension as time, Analytic Services knows that this dimension is the one on which to base the time periods for the accounts tags.

In the Sample Basic database, the Measures dimension is tagged as accounts, and the Year dimension is tagged as time.

Figure 193: Sample Basic Outline Showing Accounts and Time Tags



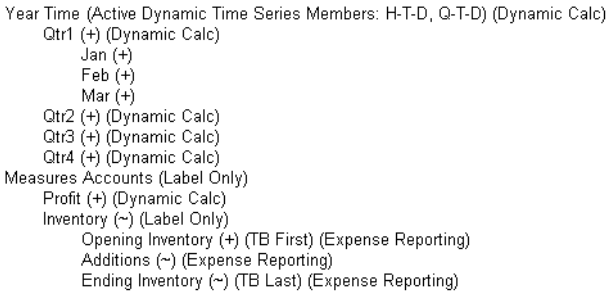
For information on tagging accounts and time dimensions, see "Creating a Time Dimension" on page 154 and "Creating an Accounts Dimension" on page 155.

Reporting the Last Value for Each Time Period

For an accounts dimension member, you can tell Analytic Services to move the last value for each time period up to the next level. To report the last value for each time period, set the member's time balance property as Last. (In the database outline, the TB Last tag is displayed.)

For example, in the Sample Basic database, the accounts member Ending Inventory consolidates the value for the last month in each quarter and uses that value for that month's parent. For example, the value for Qtr1 is the same as the value for Mar.

Figure 194: Sample Basic Outline Showing Last Tag



For information on tagging an accounts member as Last, see "Setting Time Balance Properties" on page 155.

By default, Analytic Services does not skip #MISSING or zero (0) values when calculating a parent value. You can choose to skip these values. For a discussion of how and why to skip #MISSING values, see "Skipping #MISSING and Zero Values" on page 571.

Reporting the First Value for Each Time Period

For an accounts dimension member, you can tell Analytic Services to move the first value for each time period up to the next level. To report the first value for each time period, set the member's time balance property as First. (The tag displays as TB First in the database outline.)

For example, in the Sample Basic database, the accounts member Opening Inventory consolidates the value of the first month in each quarter and uses that value for that month's parent. For example, the value for Qtr1 is the same as the value for Jan.

Figure 195: Sample Basic Outline Showing First Tag

```
Year Time (Active Dynamic Time Series Members: H-T-D, Q-T-D) (Dynamic Calc)
    Qtr1 (+) (Dynamic Calc)
        Jan (+)
        Feb (+)
        Mar (+)
    Qtr2 (+) (Dynamic Calc)
    Qtr3 (+) (Dynamic Calc)
    Qtr4 (+) (Dynamic Calc)
Measures Accounts (Label Only)
    Profit (+) (Dynamic Calc)
    Inventory (~) (Label Only)
        Opening Inventory (+) (TB First) (Expense Reporting)
        Additions (~) (Expense Reporting)
        Ending Inventory (~) (TB Last) (Expense Reporting)
```

For information on tagging an accounts member as First, see "Setting Time Balance Properties" on page 155.

By default, Analytic Services does not skip #MISSING or zero (0) values when calculating a parent value. You can choose to skip these values. For a discussion of how and why to skip #MISSING values, see "Skipping #MISSING and Zero Values" on page 571.

Reporting the Average Value for Each Time Period

For an accounts dimension member, you can tell Analytic Services to average values across time periods and consolidate the average up to the next level. For example, you can tell Analytic Services to average the values for Jan, Feb, and Mar and then use that value for the Qtr1 value. To report the average value for each time period, set the member's time balance property as Average.

For information on tagging an accounts member as Average, see "Setting Time Balance Properties" on page 155.

By default, Analytic Services does not skip #MISSING or zero (0) values when it calculates a parent value. Thus, when it calculates the average, Analytic Services aggregates the child values and divides by the number of children, regardless of whether the children have #MISSING or zero values. You can tell Analytic Services to skip #MISSING and zero values. For a discussion of how and why to skip #MISSING values, see "Skipping #MISSING and Zero Values" on page 571.

Skipping #MISSING and Zero Values

You can tell Analytic Services how to treat #MISSING and zero (0) values when doing time balance calculations. A #MISSING value is a marker in Analytic Services that indicates that the data in this location does not exist, does not contain any meaningful value, or was never entered.

By default, Analytic Services does not skip #MISSING or 0 (zero) values when calculating a parent value.

You can override this default by setting a skip property. For a discussion of how skip properties work, see "Setting Skip Properties" on page 157.

For example, if you tag an accounts dimension member as Last and Skip Missing, then Analytic Services consolidates the last non-missing child to the parent. Consider the following example:

Accounts -> Time	Jan	Feb	Mar	Qtr1
Accounts Member (**Last, Skip Missing**)	60	70	#MI	70

Tagging an account as Average and Skip Missing may produce different results from tagging that account as Average and Skip None. A calculation performed with Average and Skip None produces correct results because no data is skipped. But because grandparents with children are consolidated by summing the averages, results of a calculation on an account with Average and Skip Missing is incorrect unless you use Dynamic Calc or Two Pass tags.

Considering the Effects of First, Last, and Average Tags

The following table shows how Analytic Services consolidates the time dimension based on the time balance (TB) First, Last, and Average tags on accounts dimension members.

Accounts -> Time	Jan	Feb	Mar	Qtr1	
Accounts Member1	11	12	13	36	Value of Jan+Feb+Mar
Accounts Member2 (**TB First**)	20	25	21	20	Value of Jan

Accounts -> Time	Jan	Feb	Mar	Qtr1	
Accounts Member3 (**TB Last**)	25	21	30	30	Value of Mar
Accounts Member4 (**TB Average**)	20	30	28	26	Average of Jan, Feb, Mar

Placing Formulas on Time and Accounts Dimensions

If you place a member formula on a time or accounts dimension, it may be overwritten by a time balance calculation.

Consider the following example from Sample Basic, in which Opening Inventory is tagged as First:

Measures -> Year	Jan	Feb	Mar	Qtr1
Opening Inventory: **First**	30000	28000	27000	30000

Because Opening Inventory is tagged as First, Analytic Services calculates Opening Inventory for Qtr1 by taking the Opening Inventory for Jan value. Any member formula that is placed on Qtr1 in the database outline is overwritten by this time balance calculation.

Calculating Period-to-Date Values

You can calculate period-to-date values for data. For example, you can calculate the sales values for the current quarter up to the current month. If the current month is May, using a standard calendar quarter, the quarter total is the total of the values for April and May.

In Analytic Services, you can calculate period-to-date values in two ways:

- During a batch calculation, using the @PTD function
- Dynamically, when a user requests the values, using Dynamic Time Series members

This section explains how to use Dynamic Time Series members to dynamically calculate period-to-date values. Using Dynamic Time Series members is the most efficient method in almost all cases. For an example using the @PTD function to calculate period-to-date values, see "Calculating Period-to-Date Values" on page 509.

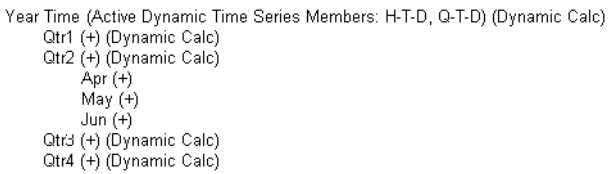
Using Dynamic Time Series Members

In order to calculate period-to-date values dynamically, you need to use a Dynamic Time Series member for a period on the dimension tagged as time. See "Specifying Accounts and Time Dimensions" on page 568.

You do not create the Dynamic Time Series member directly in the database outline. Instead, you enable a predefined Dynamic Time Series member and associate it with an appropriate generation number. This procedure creates a Dynamic Time Series member for you.

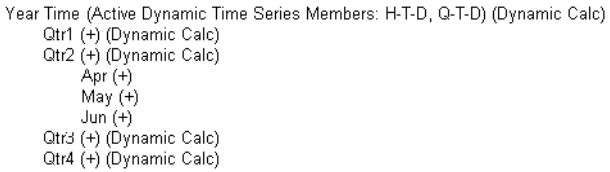
For example, if you want to calculate quarter-to-date values, you enable the Q-T-D member and associate it with the generation to which you want to apply the Dynamic Time Series member. In Sample Basic, the generation containing quarters is generation number 2, which contains the Qtr1, Qtr2, Qtr3, and Qtr4 members. Analytic Services creates a Dynamic Time Series member called Q-T-D and associates it with generation 2. The Q-T-D member calculates monthly values up to the current month in the quarter. For a brief discussion, see "Enabling Dynamic Time Series Members" on page 575.

Figure 196: Sample Basic Outline Showing Time Dimension



Dynamic Time Series members are not displayed as members in the database outline. Instead, Analytic Services lists the currently active Dynamic Time Series members in a comment on the time dimension. In the following outline, H-T-D (history-to-date) and Q-T-D (quarter-to-date) are active. H-T-D is associated with generation 1; Q-T-D is associated with generation 2.

Figure 197: Sample Basic Outline Showing Dynamic Time Series



Analytic Services provides eight predefined Dynamic Time Series members:

H-T-D	History-to-date
Y-T-D	Year-to-date
S-T-D	Season-to-date
P-T-D	Period-to-date
Q-T-D	Quarter-to-date
M-T-D	Month-to-date
W-T-D	Week-to-date
D-T-D	Day-to-date

These eight members provide up to eight levels of period-to-date reporting. How many members you use and which members you use depends on the data and the database outline.

For example, if the database contains hourly, daily, weekly, monthly, quarterly, and yearly data, you might want to report day-to date (D-T-D), week-to-date (W-T-D), month-to-date (M-T-D), quarter-to-date (Q-T-D), and year-to-date (Y-T-D) information.

If the database contains monthly data for the past 5 years, you might want to report year-to-date (Y-T-D) and history-to-date (H-T-D) information, up to a specific year.

If the database tracks data for seasonal time periods, you might want to report period-to-date (P-T-D) or season-to-date (S-T-D) information.

You can associate a Dynamic Time Series member with any generation in the time dimension except the highest generation number, irrespective of the data. For example, if you choose, you can use the P-T-D member to report quarter-to-date information. You cannot associate Dynamic Time Series members with level 0 members of the time dimension.

Note: We recommend you avoid assigning time balance properties (First, Last, Average, Skip Missing) to members set for dynamic calculations if you plan to use these members in Dynamic Time Series calculations. Doing so may retrieve incorrect values for the parent members in your accounts dimension.

Enabling Dynamic Time Series Members

To use Dynamic Time Series members, you need to enable them. If required, you can specify aliases for Dynamic Time Series members. For a brief discussion, see "Specifying Alias Names for Dynamic Time Series Members" on page 576.

➤ To enable Dynamic Time Series members, see "Enabling Dynamic Time Series Members" in the *Essbase Administration Services Online Help*.

Note: The number of generations displayed depends on the number of generations in the time dimension. You cannot associate Dynamic Time Series members with the highest generation (level 0 members).

After you enable Dynamic Time Series members in the database outline, Analytic Services adds a comment to the dimension tagged as time. Figure 198 shows the Sample Basic database with H-T-D and Q-T-D defined.

Figure 198: Sample Basic Outline Showing Dynamic Time Series Members

```
Year Time (Active Dynamic Time Series Members: H-T-D, Q-T-D) (Dynamic Calc)
```

Disabling Dynamic Time Series Members

To disable a Dynamic Time Series member, tell Analytic Services not to use the predefined member.

➤ To disable Dynamic Time Series members, see "Disabling Dynamic Time Series Members" in the *Essbase Administration Services Online Help*.

Specifying Alias Names for Dynamic Time Series Members

You can specify alias names for predefined Dynamic Time Series members, such as QtrToDate, for the Q-T-D Dynamic Time Series member. You can then use the alias names to retrieve the Dynamic Time Series members in Spreadsheet Services, Spreadsheet Add-in, or in a report.

You can create up to eight alias names for each Dynamic Time Series member. Analytic Services saves each alias name in the Dynamic Time Series alias table that you specify.

➤ To create aliases for Dynamic Time Series members, see "Creating Aliases for Dynamic Time Series Members" in the *Essbase Administration Services Online Help*.

For information on specifying and displaying alias names, see "Setting Aliases" on page 169.

Applying Predefined Generation Names to Dynamic Time Series Members

When you enable a Dynamic Time Series member and associate it with a generation number, Analytic Services creates a predefined generation name for that generation number. For information on creating generation names, see "Naming Generations and Levels" on page 175.

➤ To display generation and level names, see "Naming Generations and Levels" in the *Essbase Administration Services Online Help*.

The following table shows the Dynamic Time Series members and their corresponding generation names:

Member	Generation Name	Member	Generation Name
H-T-D	History	Q-T-D	Quarter
Y-T-D	Year	M-T-D	Month
S-T-D	Season	W-T-D	Week
P-T-D	Period	D-T-D	Day

These member and generation names are reserved for use by Analytic Services. If you use one of these generation names to create a generation name on the time dimension, Analytic Services automatically creates and enables the corresponding Dynamic Time Series member for you.

For example, in Sample Basic, you can create a generation name called Quarter for generation number 2. Quarter contains quarterly data in the members Qtr1, Qtr2, and so on. When you create the generation name Quarter, Analytic Services creates and enables a Dynamic Time Series member called Q-T-D.

Retrieving Period-to-Date Values

When you retrieve a Dynamic Time Series member, you need to tell Analytic Services the time period up to which you want to calculate the period-to-date value. This time period is known as the *latest time period* and must be a level 0 member on the time dimension.

➤ Use the following methods to specify the latest time period:

- For a specific member, in Spreadsheet Services or Spreadsheet Add-in, specify the latest period member name. Place that name after the Dynamic Time Series member or alias name. For example, Q-T-D(May) returns the quarter-to-date value by adding values for April and May.
- For a retrieval, use one of the following methods to specify the latest time period:
  - Use the <LATEST command in Report Writer.
  - Specify the Latest Time Period option in the Essbase Options dialog box in Spreadsheet Add-in or the Essbase Spreadsheet Preferences Dialog Box in Spreadsheet Services. For more information, see the relevant spreadsheet online help.

The member-specific setting—for example, Q-T-D(May)—takes precedence over the <LATEST or Latest Time Series option setting.

The following example shows Sample Basic data. Q-T-D(May) displays the period-to-date value for May that is obtained by adding the values for Apr and May (8644 + 8929 = 17573).

Figure 199: Spreadsheet Showing Period-To-Date Value for May

	Measures	Product	Market	Scenario
Qtr1	24703			
Apr	8644			
May	8929			
Jun	9534			
Qtr2	27107			
Qtr3	27912			
Qtr4	25800			
Year	105522			
Q-T-D(May)	17573			

Using Dynamic Time Series Members in Partitions

If Dynamic Time Series members are part of the shared area between databases, define the Dynamic Time Series members in both databases, just as you would for regular members. For example, if the partition definition includes Qtr1, Qtr2, Qtr3, Qtr4, and the Dynamic Time Series member Q-T-D, define the Q-T-D member in both the source database and the target database.

If a Dynamic Time Series member is *not* part of the shared area between databases, Analytic Services gets the data for that member from the source database. You do not need to define the Dynamic Time Series member in both databases. However, this configuration is generally less efficient than including the Dynamic Time Series member in the partition definition.

For a comprehensive discussion of partitioning, see Chapter 14, “Creating and Maintaining Partitions.”

Chapter 27

Developing Calculation Scripts

This chapter explains how to develop calculation scripts and how to use them to control the way Analytic Services calculates a block storage database. This chapter provides some examples of calculation scripts, which you may want to adapt for your own use. This chapter also shows you how to create and execute a simple calculation script. This chapter is not relevant to aggregate storage databases.

This chapter includes the following sections:

For a comprehensive discussion of developing formulas, which may be used in calculation scripts or in an outline, see Chapter 22, "Developing Formulas." For more examples, see Chapter 28, "Reviewing Examples of Calculation Scripts."

For information about copying, renaming, locking, and deleting calculation scripts, see "Using Analytic Services to Manage Objects" on page 962.

Understanding Calculation Scripts

A calculation script contains a series of calculation commands, equations, and formulas. You use a calculation script to define calculations other than the calculations that are defined by the database outline.

Calculation scripts are text files. Using Calculation Script Editor, you can create calculation scripts by:

- Typing the contents of the calculation script directly into the text area of the script editor
- Using the user interface features of the script editor to build the script
- Creating the script in the text editor of your choice and pasting it into Calculation Script Editor.

When you save a calculation script, it is given a `.csc` extension by default. If you run a calculation script from Essbase Administration Services or from Essbase Spreadsheet Services, it must have a `.csc` extension. However, since a calculation script is basically a text file, you can use MaxL or ESSCMD to run any text file as a calculation script.

For more information about Calculation Script Editor, see "About Calculation Script Editor" in *Essbase Administration Services Online Help*.

For example, the following calculation script calculates the Actual values in the Sample Basic database.

```
FIX (Actual)
CALC DIM(Year, Measures, Market, Product);
ENDFIX
```

You can use a calculation script to specify exactly how you want Analytic Services to calculate a database. For example, you can calculate part of a database or copy data values between members. You can design and run custom database calculations quickly by separating calculation logic from the database outline.

Analytic Services allows you to perform a default calculation (CALC ALL) or a calculation of your own choosing that you specify in a calculation script to control how Analytic Services calculates a database.

For example, you need to write a calculation script if you want to do any of the following:

- Use the FIX command to calculate a subset of a database. For more information, see "Using the FIX Command" on page 598 and the *Technical Reference*.
- Change the calculation order of the dense and sparse dimensions in a database.

- Perform a complex calculation in a specific order or perform a calculation that requires multiple iterations through the data. For example, some two-pass calculations require a calculation script.
- Perform any two-pass calculation on a dimension without an accounts tag. For a comprehensive discussion of two-pass calculation, see "Using Two-Pass Calculation" on page 1205.
- Perform a currency conversion. For a comprehensive discussion of currency conversion, see Chapter 12, "Designing and Building Currency Conversion Applications."
- Calculate member formulas that differ from formulas in the database outline. Formulas in a calculation script override formulas in the database outline.
- Use an API interface to create a custom calculation dynamically.
- Use control of flow logic in a calculation; for example, if you want to use the IF … ELSE … ENDIF or the LOOP … ENDLOOP commands.
- Clear or copy data from specific members. For an example of copying data, see "Copying Data" on page 596.
- Define temporary variables for use in a database calculation. For a discussion on how to declare temporary variables, see "Declaring Data Variables" on page 586.
- Force a recalculation of data blocks after you have changed a formula or an accounts property on the database outline.
- Control how Analytic Services uses the Intelligent Calculation feature when calculating a database. For a comprehensive discussion of intelligent calculation, see Chapter 55, "Optimizing with Intelligent Calculation."

Understanding Calculation Script Syntax

Analytic Services provides a flexible set of commands that you can use to control how a database is calculated. You can construct calculation scripts from commands and formulas. In Calculation Script Editor, the different elements of the script are color-coded to aid in script readability. You can enable auto-completion to help build scripts interactively as you type. For more information, see "About Calculation Script Editor" in *Essbase Administration Services Online Help*.

Computation, control of flow, and data declarations are discussed in the following sections. For a full list of calculation script commands, see the *Technical Reference*.

Understanding the Rules for Calculation Script Syntax

When you create a calculation script, you need to apply the following rules:

- End each formula or calculation script command with a semicolon (;), as shown in these examples.

  Example 1:

  ```
  CALC DIM(Product, Measures);
  ```

  Example 2:

  ```
  DATACOPY Plan TO Revised_Plan;
  ```

  Example 3:

  ```
  "Market Share" = Sales % Sales -> Market;
  ```

  Example 4:

  ```
  IF
    (Sales <> #MISSING) Commission = Sales * .9;
  ELSE
    Commission = #MISSING;
  ENDIF;
  ```

  You do not need to end the following commands with semicolons—IF, ENDIF, ELSE, ELSEIF, FIX, ENDFIX, LOOP, and ENDLOOP.

Although ending ENDIF statements with a semicolon (;) is not required, it is good practice to follow each ENDIF statement in a formula with a semicolon.

- Enclose a member name in double quotation marks ("") if that member name meets any of the following conditions:
  - Contains spaces; for example,

    ```
    "Opening Inventory" = "Ending Inventory" - Sales +
    Additions;
    ```

  - Is the same as an operator or function name. For a list of operator and functions names, see "Understanding the Rules for Naming Dimensions and Members" on page 143.
  - Includes any non-alphanumeric character; for example, hyphen (-), asterisk (*), and slash (/). For a complete list of special characters, see "Understanding the Rules for Naming Dimensions and Members" on page 143.
  - Contains only numerals or starts with a numeral; for example, "100" or "10Prod".
  - Begins with an ampersand (&). The leading ampersand (&) is reserved for substitution variables. If a member name begins with &, enclose it in quotation marks. Do not enclose substitution variables in quotation marks in a calculation script.
  - Contains a dot (.); for example, 1999.Jan or .100.
- If you are using an IF statement or an interdependent formula, enclose the formula in parentheses to associate it with the specified member. For example, the following formula is associated with the Commission member in the database outline:

  ```
  Commission
  (IF(Sales < 100)
     Commission = 0;
  ENDIF;)
  ```

- End each IF statement in a formula with an ENDIF statement. For example, the previous formula contains a simple IF...ENDIF statement.

- If you are using an IF statement that is nested within another IF statement, end each IF with an ENDIF statement; for example:

```
"Opening Inventory"
(IF (@ISMBR(Budget))
    IF (@ISMBR(Jan))
    "Opening Inventory" = Jan;
    ELSE
    "Opening Inventory" = @PRIOR("Ending Inventory");
    ENDIF;
ENDIF;)
```

- You do not need to end ELSE or ELSEIF statements with ENDIF statements; for example:

```
Marketing
(IF (@ISMBR(@DESCENDANTS(West)) OR
@ISMBR(@DESCENDANTS(East)))
    Marketing = Marketing * 1.5;
ELSEIF(@ISMBR(@DESCENDANTS(South)))
    Marketing = Marketing * .9;
ELSE Marketing = Marketing * 1.1;
ENDIF;)
```

  Note: If you use ELSE IF (with a space in between) rather than ELSEIF (one word) in a formula, you must supply an ENDIF for the IF statement.

- End each FIX statement with an ENDFIX statement; for example:

```
FIX(Budget,@DESCENDANTS(East))
CALC DIM(Year, Measures, Product);
ENDFIX
```

When you write a calculation script, you can use the Calculation Script Editor syntax checker to check the syntax. For a brief discussion of the syntax checker, see "Checking Syntax" on page 605.

Note: For detailed information on calculation script syntax, see the *Technical Reference*.

Understanding Calculation Commands

You can use the following calculation commands to perform a database calculation that is based on the structure and formulas in the database outline.

Note: For a complete list of calculation commands and syntax, see the *Technical Reference*.

Calculation	Command
The entire database, based on the outline	CALC ALL
A specified dimension or dimensions	CALC DIM
All members tagged as two-pass on the dimension tagged as accounts	CALC TWOPASS
The formula applied to a member in the database outline, where *membername* is the name of the member to which the formula is applied	*membername*
All members tagged as Average on the dimension tagged as accounts	CALC AVERAGE
All members tagged as First on the dimension tagged as accounts	CALC FIRST
All members tagged as Last on the dimension tagged as accounts	CALC LAST
Currency conversions	CCONV

Controlling the Flow of Calculations

You can use the following commands to manipulate the flow of calculations. For detailed information on these commands, see the *Technical Reference*.

Calculation	Commands
Calculate a subset of a database	FIX … ENDFIX
Specify the number of times that commands are iterated	LOOP … ENDLOOP

You can also use the IF and ENDIF commands to specify conditional calculations.

Note: You cannot branch from one calculation script to another calculation script.

Declaring Data Variables

You can use the following commands to declare temporary variables and, if required, to set their initial values. *Temporary variables* store the results of intermediate calculations.

You can also use substitution variables in a calculation script. For a discussion where, why, and how to use substitution variables, see "Using Substitution Variables in Calculation Scripts" on page 594.

Calculation	Commands
Declare one-dimensional array variables	ARRAY
Declare a temporary variable that contains a single value	VAR

For detailed information on the these commands, see the *Technical Reference*.

Values stored in temporary variables exist only while the calculation script is running. You cannot report on the values of temporary variables.

Variable and array names are character strings that contain any of the following characters:

- Alphabetic letters: a through z
- Numerals: 0 through 9
- Special characters: $(dollar sign), # (pound sign), and _ (underscore)

Typically, arrays are used to store variables as part of a member formula. The size of the array variable is determined by the number of members in the corresponding dimension. For example, if the Scenario dimension has four members, the following command creates an array called `Discount` with four entries. You can use more than one array at a time.

```
ARRAY Discount[Scenario];
```

Specifying Global Settings for a Database Calculation

You can use the following commands to define calculation behavior. For a detailed discussion of each specific command, see the *Technical Reference*.

Calculation	Command
To specify how Analytic Services treats #MISSING values during a calculation	SET AGGMISSG
To adjust the default calculator cache size	SET CACHE
To enable parallel calculation (see "Using Parallel Calculation" on page 1182)	SET CALCPARALLEL
To increase the number of dimensions used to identify tasks for parallel calculation (see "Using Parallel Calculation" on page 1182)	SET CALCTASKDIMS
To optimize the calculation of sparse dimension formulas in large database outlines (see "Optimizing Formulas on Sparse Dimensions in Large Database Outlines" on page 1196)	SET FRMLBOTTOMUP
To display messages to trace a calculation.	SET MSG SET NOTICE
To turn on and turn off Intelligent Calculation (see "Turning Intelligent Calculation On and Off" on page 1225)	SET UPDATECALC
To control how Analytic Services marks data blocks for the purpose of Intelligent Calculation (see "Using the SET CLEARUPDATESTATUS Command" on page 1227)	SET CLEARUPDATESTATUS
To specify the maximum number of blocks that Analytic Services can lock concurrently when calculating a sparse member formula	SET LOCKBLOCK
To turn on and turn off the Create Blocks on Equation setting. This setting controls creation of blocks when you assign non-constant values to members of a sparse dimension (see "Non-Constant Values Assigned to Members in a Sparse Dimension" on page 1197	SET CREATEBLOCKEQ

Calculation	Command
To enable calculations on potential data blocks and save these blocks when the result is not #MISSING.	SET CREATENONMISSINGBLK
For currency conversions, to restrict consolidations to parents that have the same defined currency (see "Calculating Databases" on page 227)	SET UPTOLOCAL

A SET command in a calculation script stays in effect until the next occurrence of the same SET command.

Consider the following calculation script:

```
SET MSG DETAIL;
CALC DIM(Year);
SET MSG SUMMARY;
CALC DIM(Measures);
```

Analytic Services displays messages at the detail level when calculating the Year dimension. However, when calculating the Measures dimension, Analytic Services displays messages at the summary level.

Some SET calculation commands trigger additional passes through the database. Consider this calculation script:

```
SET AGGMISSG ON;
Qtr1;
SET AGGMISSG OFF;
East;
```

Analytic Services calculates member combinations for Qtr1 with SET AGGMISSG turned on. Analytic Services then does a second calculation pass through the database and calculates member combinations for East with SET AGGMISSG turned off. For more information on the setting for consolidating missing values, see the SET AGGMISSG command in the *Technical Reference*. For more information on calculation passes, see "Using Two-Pass Calculation" on page 1205.

Adding Comments

You can include comments to annotate calculation scripts. Analytic Services ignores these comments when it runs the calculation script.

To include a comment, start the comment with /* and end the comment with */. Consider the following comment:

```
/*   This is a calculation script comment
     that spans two lines.*/
```

Planning Calculation Script Strategy

You can type a calculation script directly into the text area of Calculation Script Editor, or you can use the user interface features of Calculation Script Editor to build the calculation script.

Using Formulas in a Calculation Script

You can place member formulas in a calculation script. When you place formulas in a calculation script, they override any conflicting formulas that are applied to members in the database outline.

In a calculation script, you can perform both of the following operations:

- Calculate a member formula on the database outline
- Define a formula

To calculate a formula that is applied to a member in the database outline, simply use the member name followed by a semicolon (;); for example:

```
Variance;
```

This command calculates the formula applied to the Variance member in the database outline.

To override values that result from calculating an outline, manually apply a formula that you define in a calculation script, using Calculation Script Editor or by creating a `.txt` file. The following formula cycles through the database, adding the values in the members Payroll, Marketing, and Misc and placing the result in the Expenses member. This formula overrides any formula placed on the Expenses member in the database outline.

```
Expenses = Payroll + Marketing + Misc;
```

Note: You cannot apply formulas to shared members or label only members.

For more information about formulas, see Chapter 22, “Developing Formulas.”

Basic Equations

You can use equations in a calculation script to assign value to a member, as follows:

```
Member = mathematical expression;
```

In this equation, *Member* is a member name from the database outline, and *mathematical expression* is any valid mathematical expression. Analytic Services evaluates the expression and assigns the value to the specified member.

For example, the following formula causes Analytic Services to cycle through the database, subtracting the values in COGS from the values in Sales and placing the result in Margin:

```
Margin = Sales - COGS;
```

To *cycle through* a database means that Analytic Services takes a calculation pass through the database. For more information on calculation passes, see "Using Two-Pass Calculation" on page 1205.

The next formula cycles through the database subtracting the values in Cost from the values in Retail, calculating the resulting values as a percentage of the values in Retail, and placing the results in Markup:

```
Markup = (Retail - Cost) % Retail;
```

You can also use the > (greater than) and < (less than) logical operators in equations; for example:

```
Sales Increase Flag = Sales->Feb > Sales->Jan;
```

If it is true that February sales are greater than January sales, Sales Increase Flag results in a 1 value; if false, the result is a 0 value.

Conditional Equations

When you use an IF statement as part of a member formula in a calculation script, you need to perform both of the following tasks:

- Associate the IF statement with a single member
- Enclose the IF statement in parentheses

A sample IF statement is illustrated in the following example:

```
Profit
(IF (Sales > 100)
   Profit = (Sales - COGS) * 2;
ELSE
  Profit = (Sales - COGS) * 1.5;
ENDIF;)
```

Analytic Services cycles through the database and performs the following calculations:

1. The IF statement checks to see if the value of Sales for the current member combination is greater than 100.
2. If Sales is greater than 100, Analytic Services subtracts the value in COGS from the value in Sales, multiplies the difference by 2, and places the result in Profit.

3. If Sales is less than or equal to 100, Analytic Services subtracts the value in COGS from the value in Sales, multiplies the difference by 1.5, and places the result in Profit.

The whole of the IF … ENDIF statement is enclosed in parentheses and associated with the Profit member, `Profit(IF(...)...)`.

Interdependent Formulas

When you use an interdependent formula in a calculation script, the same rules apply as for the IF statement. You need to perform both of the following tasks:

- Associate the formula with a single member
- Enclose the formula in parentheses

Consider the interdependent formula discussed earlier. If you place the formula in a calculation script, you enclose the whole formula in parentheses and associate it with the Opening Inventory member, as follows:

```
"Opening Inventory"
(IF(NOT @ISMBR (Jan))"Opening Inventory" =
    @PRIOR("Ending Inventory"));
    ENDIF;
"Ending Inventory" = "Opening Inventory" - Sales + Additions;)
```

Using a Calculation Script to Control Intelligent Calculation

Assume that you have a formula on a sparse dimension member and the formula contains either of the following:

- A relationship function (for example, @PRIOR or @NEXT)
- A financial function (for example, @NPV or @INTEREST)

Analytic Services always recalculates the data block that contains the formula, even if the data block is marked as clean for the purposes of Intelligent Calculation. For more information, see “Calculating Data Blocks” on page 1231. For more information about Intelligent Calculation, see Chapter 55, “Optimizing with Intelligent Calculation.”

Grouping Formulas and Calculations

You may achieve significant calculation performance improvements by carefully grouping formulas and dimensions in a calculation script. For a discussion of the appropriate use of parentheses and for examples, see "Calculating a Series of Member Formulas" on page 593 and "Calculating a Series of Dimensions" on page 594.

When you run a calculation script, Analytic Services automatically displays the calculation order of the dimensions for each pass through the database so that you can tell how many times Analytic Services has cycled through the database during the calculation. Analytic Services writes these information messages in the application log. The messages can also be viewed during your session in the following ways:

- In the Messages Panel of Administration Services Console
- In the standard output (command-line window) of MaxL Shell or ESSCMD

➤ To display the application log, see "Viewing the Analytic Server and Application Logs" on page 997.

Calculating a Series of Member Formulas

When you calculate formulas, avoid using parentheses unnecessarily. The following formulas cause Analytic Services to cycle through the database once, calculating both formulas in one pass:

```
Profit = (Sales - COGS) * 1.5;
Market = East + West;
```

Similarly, the following configurations cause Analytic Services to cycle through the database only once, calculating the formulas on the members Qtr1, Qtr2, and Qtr3:

```
Qtr1;
Qtr2;
Qtr3;
```

or

```
(Qtr1;
Qtr2;
Qtr3;)
```

However, the inappropriately placed parentheses in the following example causes Analytic Services to perform two calculation passes through the database, once calculating the formulas on the members Qtr1 and Qtr2 and once calculating the formula on Qtr3:

```
(Qtr1;
Qtr2;)
Qtr3;
```

Calculating a Series of Dimensions

When you calculate a series of dimensions, you can optimize performance by grouping the dimensions wherever possible.

For example, the following formula causes Analytic Services to cycle through the database only once:

```
CALC DIM(Year, Measures);
```

However, the following syntax causes Analytic Services to cycle through the database twice, because Analytic Services cycles through once for each CALC DIM command:

```
CALC DIM(Year);
CALC DIM(Measures);
```

Using Substitution Variables in Calculation Scripts

You can use substitution variables in calculation scripts. Substitution variables are useful, for example, when you reference information or lists of members that change frequently.

When you include a substitution variable in a calculation script, Analytic Services replaces the substitution variable with the value you specified for the substitution variable.

You create and specify values for substitution values in Essbase Administration Services. For a comprehensive discussion of substitution variables, see "Using Substitution Variables" on page 133.

You can create variables at the server, application, and database levels. When you use a substitution variable in a calculation script, it must be available to the calculation script. For example, if you create a substitution variable at the database

level, it is only available to calculation scripts within the database. However, if you create a variable at the server level, it is available to any calculation script on the server.

Add the ampersand (&) character before a substitution variable in a calculation script. Analytic Services treats any string that begins with a leading ampersand as a substitution variable, replacing the variable with its assigned value before parsing the calculation script.

For example, `&CurQtr;` becomes `Qtr1;` if you have given the substitution variable `&CurQtr` the value `Qtr1`.

Consider an example in which you want to calculate Sample Basic data for the current quarter. You can use the following calculation script to perform this calculation:

```
FIX(&CurQtr)
CALC DIM(Measures, Product);
ENDFIX
```

You then define the substitution variable `CurQtr` as the current quarter; for example, Qtr3. Analytic Services replaces the variable CurQtr with the value Qtr3 when it runs the calculation script.

Clearing Data

You can use the following commands to clear data. If you want to clear an entire database, see "Clearing Data" in the *Essbase Administration Services Online Help*.

Calculation	Command
Changes the values of the cells you specify to #MISSING. The data blocks are not removed. You can use the FIX command with the CLEARDATA command to clear a subset of a database.	CLEARDATA
Removes the entire contents of a block, including all the dense dimension members. Analytic Services removes the entire block, unless CLEARBLOCK is inside a FIX command on members within the block.	CLEARBLOCK
Removes blocks for Dynamic Calc and Store member combinations. For more information, see Chapter 25, "Dynamically Calculating Data Values."	CLEARBLOCK DYNAMIC

The following examples are based on the Sample Basic database. If the Scenario dimension is dense, the following example removes all the data cells that do not contain input data values and intersect with member Actual from the Scenario dimension.

```
FIX(Actual)
CLEARBLOCK NONINPUT;
ENDFIX
```

If the Scenario dimension is sparse, the following formula removes only the blocks whose Scenario dimension member is Actual. The other blocks remain:

```
FIX(Actual)
CLEARBLOCK NONINPUT;
ENDFIX
```

For example, the following formula clears all the Actual data values for Colas:

```
CLEARDATA Actual -> Colas;
```

Copying Data

You can use the DATACOPY calculation command to copy data cells from one range of members to another range of members in a database. The two ranges must be the same size.

For example, in the Sample Basic database, the following formula copies Actual values to Budget values:

```
DATACOPY Actual TO Budget;
```

You can use the FIX command to copy a subset of values.

For example, in the Sample Basic database, the following formula copies Actual values to Budget values for the month of January only:

```
FIX (Jan)
DATACOPY Actual TO Budget;
ENDFIX
```

For more information about the DATACOPY command, see the *Technical Reference*. For more information about the FIX command, see “Using the FIX Command” on page 598.

Calculating a Subset of a Database

➤ To calculate a subset of a database, use either of the following methods:

- Create a formula using member set functions to calculate lists of members.
- Use the FIX … ENDFIX commands to calculate a range of values.

For information about using member lists, see "Calculating Lists of Members" on page 597. For examples of use of the FIX command, see "Using the FIX Command" on page 598.

Note: When you have Intelligent Calculation turned on, the newly calculated data blocks are not marked as clean after a partial calculation of a database. When you calculate a subset of a database, you can use the SET CLEARUPDATESTATUS AFTER command to ensure that the newly calculated blocks are marked as clean. Using this command ensures that Analytic Services recalculates the database as efficiently as possible using Intelligent Calculation. For a comprehensive discussion of Intelligent Calculation, see Chapter 55, "Optimizing with Intelligent Calculation."

Calculating Lists of Members

You can use a member set function to generate a list of members that is based on a member you specify. For example, you can use the @IDESCENDANTS function to generate a list of all the descendants of a specified member.

In the Sample Basic database, `@IDESCENDANTS("Total Expenses");` generates the following list of members—Total Expenses, Marketing, Payroll, and Misc.

When you use a member set function in a formula, Analytic Services generates a list of members before calculating the formula.

For detailed information on these and other member set functions, see the *Technical Reference*.

Using the FIX Command

The FIX … ENDFIX commands are particularly useful to calculate a carefully defined subset of the values in a database. For example, the following calculation script calculates only the Budget values for only the descendants of East (New York, Massachusetts, Florida, Connecticut, and New Hampshire) in the Sample Basic database:

```
FIX(Budget,@DESCENDANTS(East))
CALC DIM(Year, Measures, Product);
ENDFIX
```

The next example fixes on member combinations for the children of East that have a user-defined attribute (UDA) of New Mkt. For information on defining UDAs, see Chapter 8, “Creating and Changing Database Outlines.”

```
FIX(@CHILDREN(East) AND @UDA(Market,"New Mkt"))
Marketing = Marketing * 1.1;
ENDFIX
```

The next example uses a wildcard match to fix on member names that end in the characters -10. In Sample Basic, this example fixes on the members 100-10, 200-10, 300-10, and 400-10.

```
FIX(@MATCH(Product, "???-10"))
Price = Price * 1.1;
ENDFIX
```

When you use the FIX command *only* on a dense dimension, Analytic Services retrieves the entire block that contains the required value or values for the member or members that you specify. Thus, I/O is not affected, and the calculation performance time is improved.

When you use the FIX command on a sparse dimension, Analytic Services retrieves the block for the specified sparse dimension member or members. Thus, I/O may be greatly reduced.

Analytic Services cycles through the database once for each FIX command that you use on dense dimension members. When possible, combine FIX blocks to improve calculation performance. For example, the following calculation script causes Analytic Services to cycle through the database only once, calculating both the Actual and the Budget values:

```
FIX(Actual,Budget)
CALC DIM(Year, Measures);
ENDFIX
```

However, this calculation script causes Analytic Services to cycle through the database twice, once calculating the Actual data values and once calculating the data values for Budget:

```
FIX(Actual)
CALC DIM(Year, Measures);
ENDFIX
FIX(Budget)
CALC DIM(Year, Measures);
ENDFIX
```

You cannot FIX on a subset of a dimension that you calculate within a FIX statement. For example, the following calculation script returns an error message because the CALC DIM operation calculates the entire Market dimension, although the FIX above it fixes on specific members of the Market dimension.

```
FIX(@CHILDREN(East) AND @UDA(Market,"New Mkt"))
CALC DIM(Year, Measures, Product, Market);
ENDFIX
```

Note: The FIX command has some restrictions. For detailed information on the restrictions and on using the FIX command, see the *Technical Reference*.

Enabling Calculations on Potential Blocks

When you use a formula on a dense member in a dense dimension, if the resultant values are from a dense dimension and the operand or operands are from a sparse dimension, Analytic Services does not automatically create the required blocks.

Consider an example from Sample Basic, in which you want to create budget sales and expense data from existing actual data. Sales and Expenses are members in the dense Measures dimension. Budget and Actual are members in the sparse Scenario dimension.

```
FIX(Budget)
      (Sales = Sales -> Actual * 1.1;
       Expenses = Expenses -> Actual * .95;)
ENDFIX
```

Note that Sales and Expenses, the results of the equations, are dense dimension members, and the operand, Actual, is in a sparse dimension. Because Analytic Services executes dense member formulas only on existing data blocks, the calculation script above does *not* create the required data blocks and Budget data values are not calculated for blocks that do not already exist.

You can solve the problem using the following techniques, each with its own advantages and disadvantages:

- "Using DATACOPY to Copy Existing Blocks" on page 600
- "Using SET CREATENONMISSINGBLK to Calculate All Potential Blocks" on page 601

Using DATACOPY to Copy Existing Blocks

You can use the DATACOPY command to create a new block for each existing block, and then perform calculations on the new blocks, as shown in the following example:

```
DATACOPY Sales -> Actual TO Sales -> Budget;
DATACOPY Expenses -> Actual TO Expenses -> Budget;
FIX(Budget)
      (Sales = Sales -> Actual * 1.1;
       Expenses = Expenses -> Actual *  .95;)
ENDFIX
```

Analytic Services creates blocks that contain the Budget values for each corresponding Actual block that already exists. After the DATACOPY commands are finished, the remaining part of the script changes the values.

Using DATACOPY works well in the following circumstances:

- There is a mathematical relationship between values in existing blocks and their counterparts created by the DATACOPY. For example, in the preceding example, the Budget values can be calculated based on the existing Actual values.
- None of the blocks that are copied contain only #MISSING values. It is possible that blocks will be written that contain only #MISSING values. Unneeded #MISSING blocks require Analytic Services resource and processing time.

CAUTION: DATACOPY creates the new blocks with identical values in ALL cells from the source blocks. If the formula only performs on a portion of the block, these copied cells will remain at the end of calculation, potentially resulting in unwanted values.

Using SET CREATENONMISSINGBLK to Calculate All Potential Blocks

If you are concerned about unwanted values, instead of using the DATACOPY approach described above, you can use the SET CREATENONMISSINGBLK ON calculation command. The SET CREATENONMISSINGBLK ON calculation command calculates all potential blocks in memory and then stores *only* the calculated blocks that contain data values.

The following example script demonstrates using the SET CREATENONMISSINGBLK ON calculation command to create budget sales and expense data from existing actual data. Sales and Expenses are members in the dense Measures dimension. Budget and Actual are members in the sparse Scenario dimension.

```
FIX(Budget)
SET CREATENONMISSINGBLK ON
      (Sales = Sales -> Actual * 1.1;
       Expenses = Expenses -> Actual * .95;)
ENDFIX
```

The SET CREATENONMISSINGBLK calculation command can be useful when calculating values on dense or sparse dimensions. For additional information about this command, see the *Technical Reference*.

Note: If the Create Blocks on Equations setting for sparse dimensions is ON, the SET CREATENONMISSINGBLK ON temporarily overrides the Create Blocks on Equations setting until a SET CREATENONMISSINGBLK OFF command is encountered or the calculation script is completed. For more information about the Create Blocks on Equations setting, see "Non-Constant Values Assigned to Members in a Sparse Dimension" on page 1197 or the SET CREATEBLOCKONEQ calculation command in the *Technical Reference*.

The advantage to using the SET CREATENONMISSINGBLK command is that, when applied on dense members, only data cells that are affected by the member formula are saved. The disadvantage is that too many potential blocks may be materialized in memory, possibly affecting calculation performance. When you use this command, limit the number of potential blocks; for example, by using FIX to restrict the scope of the blocks to be calculated.

For additional information about using the SET CREATENONMISSINGBLK ON calculation command, see the *Technical Reference*.

Writing Calculation Scripts for Partitions

A partitioned application can span multiple servers, processors, or computers. For a comprehensive discussion of partitioning, see Chapter 13, "Designing Partitioned Applications" and Chapter 14, "Creating and Maintaining Partitions."

You can achieve significant calculation performance improvements by partitioning applications and running separate calculations on each partition.

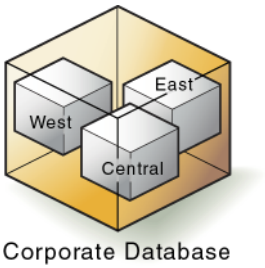
However, when you use partitioning, you need to perform both of the following tasks:

- Consider carefully the performance impact on the overall database calculation. You might choose to use any of the following methods to improve performance:
  - Redesign the overall calculation to avoid referencing remote values that are in a transparent partition in a remote database.
  - Dynamically calculate a value in a remote database. See "Dynamically Calculating Data in Partitions" on page 565.
  - Replicate a value in the database that contains the applicable formula. For example, if you are replicating quarterly data for the Eastern region, replicate only the values for Qtr1, Qtr2, Qtr3, and Qtr4, and calculate the parent Year values locally.
- Ensure that a referenced value is up-to-date when Analytic Services retrieves it. Choose one of the options previously discussed (redesign, dynamically calculate, or replicate) or calculate the referenced database before calculating the formula.

Controlling Calculation Order for Partitions

You need to calculate databases in a specific order to ensure that Analytic Services calculates the required results. For example, consider the following partitions in which you view information from the West, Central, and East databases transparently from the Corporate database.

Figure 200: Calculating Partitions



West, Central, and East contain only actual values. Corporate contains actual and budgeted values. Although you can view the West, Central, and East data in the Corporate database, the data exists only in the West, Central, and East databases; it is not duplicated in the Corporate database.

Therefore, when Analytic Services calculates Corporate, it needs to take the latest values from West, Central, and East. To obtain the required results, you need to calculate West, Central, and East before you calculate Corporate.

Reviewing the Process for Creating Calculation Scripts

Use this process to create a calculation script:

1. Create a new calculation script or open an existing calculation script.

   See “Creating Scripts” or “Opening Scripts” in *Essbase Administration Services Online Help*.

2. Enter or edit the contents of the calculation scripts.

   In the *Essbase Administration Services Online Help*, see "About Calculation Script Editor." This may include some or all of the following tasks:

   - Associating a script with an outline
   - Searching an outline tree for members
   - Inserting dimensions, members, and aliases in a script from an outline tree
   - Inserting functions and commands in a script from a tree
   - Using syntax auto-completion
   - Checking script syntax
   - Executing scripts
   - Viewing color-coded script elements
   - Searching for text in a script
   - Changing fonts

3. Validate the calculation script.

   See "Checking Syntax" on page 605 and "Checking Script Syntax" in the *Essbase Administration Services Online Help*.

4. Save the calculation script.

   See "Saving Calculation Scripts" on page 605 and "Saving Scripts" in the *Essbase Administration Services Online Help*.

5. Execute the calculation script.

   See "Executing Calculation Scripts" on page 606, "Checking the Results of Calculations" on page 607, and "Executing Calculation Scripts" in the *Essbase Administration Services Online Help*.

6. If necessary, perform other operations on the calculation script.

   In the *Essbase Administration Services Online Help*, see the following topics:

   - "Locking and Unlocking Objects"
   - "Copying Scripts"
   - "Renaming Scripts"

- “Deleting Scripts”
- “Printing Scripts”

For more information on calculation scripts, see “Understanding Calculation Scripts” on page 579.

Checking Syntax

Analytic Services includes a syntax checker that tells you about any syntax errors in a calculation script. For example, Analytic Services tells you if you have typed a function name incorrectly. The syntax checker cannot tell you about semantic errors in a calculation script. Semantic errors occur when a calculation script does not work as you expect. To find semantic errors, always run the calculation, and check the results to ensure they are as you expect.

Analytic Services displays the syntax checker results in the messages panel in Essbase Administration Services Console. If Analytic Services finds no syntax errors, that is indicated in the messages panel. One error is displayed at a time.

If Analytic Services finds one or more syntax errors, it usually displays the number of the line that includes the error and a brief description of the error. For example, if you do not include a semicolon end-of-line character at the end of a calculation script command, Analytic Services displays a message similar to the following:

```
Error: line 1: invalid statement; expected semicolon
```

When you reach the first or last error, Analytic Services displays the following message:

```
No more errors
```

➤ To check the syntax of a calculation script in Calculation Script Editor, see “Checking Script Syntax” in the *Essbase Administration Services Online Help*.

Saving Calculation Scripts

Calculation scripts created using Administration Services are given a `.csc` extension by default. If you run a calculation script from Administration Services or from Essbase Spreadsheet Services, it must have a `.csc` extension. However, a calculation script is a text file, and you can use MaxL or ESSCMD to run any text file as a calculation script.

A calculation script can also be a string defined in memory. You can access this string via the API on an Analytic Services client or an Analytic Server. Thus, from dialog boxes, you can dynamically create a calculation script that is based on user selections.

You can save a calculation script in the following locations:

- As a file on a client machine.
- As an object on an Analytic Server. If you want other users to have access to the calculation script, save it on an Analytic Server. You can associate the script with the following objects:
  - An application and all the databases within the application, which lets you run the script against any database in the application. Calculation scripts associated with an application are saved in the /*ARBORPATH*/app/*appname* directory on the Analytic Server computer. *ARBORPATH* is the Analytic Services install directory, and *appname* is the application with which you have associated the calculation script.
  - A database, which lets you run the script against the specified database. Calculation scripts associated with a database are saved in the /*ARBORPATH*/app/*appname* /*dbname* directory on the Analytic Server computer. *ARBORPATH* is the Analytic Services install directory, *appname* is the application containing the database, and *dbname* is the database with which you have associated the calculation script.

➤ To save a calculation script using Calculation Script Editor, see “Saving Scripts” in *Essbase Administration Services Online Help*.

Executing Calculation Scripts

Before you can execute a calculation script in Administration Services, you must save it as an object on the Analytic Server, on a client computer, or on a network. See “Saving Calculation Scripts” on page 605.

When you use Administration Services to execute a calculation script, you can execute the calculation in the background so that you can continue working as the calculation processes. You can then check the status of the background process to see when the calculation has completed. For more information, see “Executing Calculation Scripts” in *Essbase Administration Services Online Help*.

➤ To execute a calculation script, use any of the following methods:

Tool	Topic	Location
Administration Services	Executing Calculation Scripts	*Essbase Administration Services Online Help*
MaxL	**execute calculation**	*Technical Reference*
ESSCMD	RUNCALC	*Technical Reference*
Essbase Spreadsheet Add-in	ESSBASE > CALCULATION	*Essbase Spreadsheet Add-in User's Guide*
Essbase Spreadsheet Services	Calculating with a Script	*Essbase Spreadsheet Services Online Help*

Checking the Results of Calculations

After you execute a calculation script, you can check the results of the calculation in the Spreadsheet Add-in or other appropriate tool.

When you execute a calculation using Administration Services or Spreadsheet Services, you can view the calculation messages in the application log. When you use ESSCMD to execute a calculation script, you can view the messages in the ESSCMD window. When you use MaxL or ESSCMD to execute a calculation script, Analytic Services displays the messages to the standard output (command-line window), depending on the level of messages you have set to display in MaxL Shell or ESSCMD.

➤ To display the application log, see "Viewing the Analytic Server and Application Logs" on page 997.

Analytic Services provides the following information:

- The calculation order of the dimensions for each pass through the database
- The total calculation time

You can use these messages to understand how the calculation is performed and to tune it for the next calculation. To display more detailed information, you can use the SET MSG SUMMARY, SET MSG DETAIL, and SET NOTICE commands in a calculation script. For more information, see "Specifying Global Settings for a Database Calculation" on page 587.

Copying Calculation Scripts

You can copy calculation scripts to applications and databases on any Analytic Server, according to your permissions. You can also copy scripts across servers as part of application migration.

➤ To copy a calculation script, use any of the following methods:

Tool	Topic	Location
Administration Services	Copying Scripts	*Essbase Administration Services Online Help*
MaxL	**create calculation as**	*Technical Reference*
ESSCMD	COPYOBJECT	*Technical Reference*

Chapter 28

Reviewing Examples of Calculation Scripts

The examples in this chapter illustrate different types of calculation scripts, which you may want to adapt for your own use.

This chapter includes the following examples:

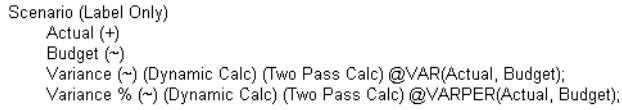
Note: Since you do not use calculation scripts with aggregate storage databases, the information in this chapter is not relevant them.

For examples that use the Intelligent Calculation commands SET UPDATECALC and SET CLEARUPDATESTATUS in calculation scripts, see "Reviewing Examples That Use SET CLEARUPDATESTATUS" on page 1229 and "Reviewing Examples and Solutions for Multiple-Pass Calculations" on page 1235.

Calculating Variance

The Sample Basic database includes a calculation of the percentage of variance between Budget and Actual values.

Figure 201: Calculating Variance and Variance %



During a default calculation of the Sample Basic database, Analytic Services aggregates the values on the Market and Product dimensions. Percentage values do not aggregate correctly. Therefore, the Variance % formula needs to be recalculated after the default calculation.

In the Sample Basic outline, Variance % is tagged as a Dynamic Calc, two-pass member. Thus, Analytic Services dynamically calculates Variance % values when they are retrieved. The dynamic calculation overwrites the incorrect values with the correctly calculated percentages. If you choose not to tag Variance % as a Dynamic Calc, two-pass member, use the following calculation script to recalculate Variance %. For a comprehensive discussion of Dynamic Calc members, see Chapter 25, "Dynamically Calculating Data Values." See "Using Two-Pass Calculation" on page 1205 for information about calculation of two-pass members.

Assuming that Intelligent Calculation is turned on (the default), the following calculation script performs a default calculation and then recalculates the formula on Variance %:

```
CALC ALL;
SET UPDATECALC OFF;
SET CLEARUPDATESTATUS AFTER;
"Variance %";
```

Analytic Services performs the following actions:

1. Analytic Services uses the CALC ALL command to perform a default calculation of the database.

   Note: Alternatively, run a default calculation of the database outline without using a calculation script.

2. The SET UPDATECALC OFF command turns off Intelligent Calculation.

3. The CLEARUPDATESTATUS AFTER command tells Analytic Services to mark the calculated blocks calculated by the variance formula of the calculation script as clean, even though the variance calculation is a partial calculation of the database (by default, data blocks are marked as clean only after a full calculation of the database).
4. Analytic Services cycles through the database calculating the formula for Variance %.

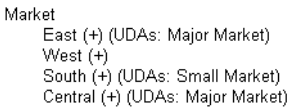
For information on calculating *statistical* variance, see the *Technical Reference*.

For information on using a calculation script for two-pass calculations, see "Choosing Two-Pass Calculation Tag or Calculation Script" on page 1210. For a comprehensive discussion on developing and using formulas to calculate a database, see Chapter 22, "Developing Formulas."

Calculating Database Subsets

In this example, based on the Sample Basic database, the Marketing managers of the regions East, West, South, and Central need to calculate their respective areas of the database.

Figure 202: Market Dimension from the Sample Basic Database



The marketing manager of the region East uses the following calculation script to calculate the data values for East:. Notice how @DESCENDENTS(East) is used to limit the calculations to the Eastern region.

```
/* Calculate the Budget data values for the descendants of East */
FIX(Budget, @DESCENDANTS(East))
CALC DIM(Year, Measures, Product);
ENDFIX
/* Consolidate East */
FIX(Budget)
@DESCENDANTS(East);
ENDFIX
```

The script calculates the Year, Measures, and Product dimensions for each child of East.

Analytic Services performs the following actions:

1. Analytic Services fixes on the Budget values of the descendants of East.
2. The Year, Measures, and Product dimensions are calculated in one pass of the database for all Budget values of the descendants of East.
3. Analytic Services fixes on the Budget values for all members on the other dimensions.
4. Analytic Services aggregates the descendants of East and places the result in East.

Loading New Budget Values

This example calculates the Budget values of the Sample Basic database and then recalculates the Variance and Variance % members of the database:

```
/* Calculate all Budget values */
FIX(Budget)
CALC DIM(Year, Product, Market, Measures);
ENDFIX

/* Recalculate the Variance and Variance % formulas, which
      require two passes */
Variance;
"Variance %";
```

Analytic Services performs the following actions:

1. Analytic Services fixes on the Budget values.
2. Analytic Services calculates all Budget values. The CALC DIM command is used to calculate all the dimensions except for the Scenario dimension, which contains Budget.
3. Analytic Services calculates the formula applied to Variance in the database outline.
4. Analytic Services calculates the formula applied to Variance % in the database outline.

Calculating Product Share and Market Share Values

This example, based on the Sample Basic database, calculates product share and market share values for each market and each product.

The product and market share values are calculated as follows:

- Each member as a percentage of the total
- Each member as a percentage of its parent

Assume that you add four members to the Measures dimension—Market Share, Product Share, Market %, and Product %.

```
/* First consolidate the Sales values to ensure that they are
accurate */
FIX(Sales)
CALC DIM(Year, Market, Product);
ENDFIX

/* Calculate each market as a percentage of the
total market for each product */
"Market Share" = Sales % Sales -> Market;

/* Calculate each product as a percentage of the
total product for each market */
"Product Share" = Sales % Sales -> Product;

/* Calculate each market as a percentage of its
parent for each product */
"Market %" = Sales % @PARENTVAL(Market, Sales);

/* Calculate each product as a percentage its
parent for each market */
"Product %" = Sales % @PARENTVAL(Product, Sales);
```

Analytic Services performs the following actions:

1. Analytic Services fixes on the Sales values and consolidates all the Sales values. The CALC DIM command is used to calculate the Year, Market, and Product dimensions. The Measures dimension contains the Sales member and therefore is not consolidated. The Scenario dimension is label only and therefore does not need to be consolidated.
2. Analytic Services cycles through the database and calculates Market Share. It takes the Sales value for each product in each market for each month. It calculates this Sales value as a percentage of total Sales in all markets for each product (Sales -> Market).
3. Analytic Services calculates Product Share. It takes the Sales value for each product in each market for each month. It calculates this Sales value as a percentage of total Sales of all products in each market (Sales -> Product).
4. Analytic Services calculates Market %. It takes the Sales value for each product in each market for each month. It calculates this Sales value as a percentage of the Sales value of the parent of the current member on the Market dimension. It uses the @PARENTVAL function to obtain the Sales value of the parent on the Market dimension.
5. Analytic Services calculates Market %. It takes the Sales value for each product in each market for each month. It calculates this Sales value as a percentage of the Sales value of the parent of the current member on the Product dimension. It uses the @PARENTVAL function to obtain the Sales value of the parent on the Product dimension.

Allocating Costs Across Products

The following example is based on the Sample Basic database. It allocates overhead costs to each product in each market for each month.

The overhead costs are allocated based on each product's Sales value as a percentage of the total Sales for all products.

Assume that you add two members to the Measures dimension—OH_Costs for the allocated overhead costs and OH_TotalCost for the total overhead costs.

```
/* Declare a temporary array called ALLOCQ
based on the Year dimension */
ARRAY ALLOCQ[Year];

/*Turn the Aggregate Missing Values setting off.
```

```
If this is your system default, omit this line */
SET AGGMISSG OFF;

/* Allocate the overhead costs for Actual values */
FIX(Actual)
OH_Costs (ALLOCQ=Sales/Sales->Product; OH_Costs =
OH_TotalCost->Product * ALLOCQ;);

/* Calculate and consolidate the Measures dimension */
CALC DIM(Measures);
ENDFIX
```

Analytic Services performs these calculations:

1. Analytic Services creates a one-dimensional array called ALLOCQ. The size of ALLOCQ is based on the number of members in the Year dimension. Analytic Services uses ALLOCQ to store the value of Sales as a percentage of total Sales temporarily for each member combination.
2. The `SET AGGMISSG OFF;` command means that #MISSING values are not aggregated to their parents. Data values stored at parent levels are not overwritten. If this is your system default, you can omit this line. For information on setting the default for aggregating #MISSING values, see "Consolidating #MISSING Values" on page 1217.

   - Analytic Services fixes on the Actual values.
   - Analytic Services cycles through the member combinations for Actual and calculates OH_Costs.
   - It then takes the Sales value for each product in each market for each month. It calculates this Sales value as a percentage of total Sales for all products in each market (Sales -> Product). It places the result in ALLOCQ.
   - It then takes the total overhead costs for all products (OH_TotalCost -> Product) and multiplies it by the value it has just placed in ALLOCQ. It places the result in OH_Costs.

   Notice that both of the equations are enclosed in parentheses () and associated with the OH_Costs member, OH_Costs(equation1; equation2;).
3. Analytic Services calculates and consolidates the Measures dimension.

Allocating Values Within or Across Dimensions

Using the @ALLOCATE and @MDALLOCATE functions, you can allocate values to members in the same dimension or to members in multiple dimensions.

Allocating Within a Dimension

The following example uses the @ALLOCATE function to allocate budgeted total expenses across expense categories for two products. The budgeted total expenses are allocated based on the actual values for the prior year.

The following example is based on the Sample Basic database. Assume that you have made the following changes to Sample Basic:

- Added a child, Lease, under Total Expenses in the Measures dimension
- Added a child, PY Actual, to the Scenario dimension
- Removed the Dynamic Calc tag from the Total Expenses member

Figure 203: Modified Measures and Scenario Dimensions from the Sample Basic Database

```
Measures Accounts (Label Only)
      Profit (+) (Dynamic Calc)
            Margin (+) (Dynamic Calc)
            Total Expenses (-) (Expense Reporting)
                  Lease (+)
                  Marketing (+) (Expense Reporting)
                  Payroll (+) (Expense Reporting)
                  Misc (+) (Expense Reporting)
      Inventory (~) (Label Only)
      Ratios (~) (Label Only)
Product {Caffeinated, Intro Date, Ounces, Pkg Type }
Market {Population }
Scenario (Label Only)
      Actual (+)
      Budget (~)
      PY Actual (+)
      Variance (~) (Dynamic Calc) (Two Pass Calc) @VAR(Actual, Budget);
      Variance % (~) (Dynamic Calc) (Two Pass Calc) @VARPER(Actual, Budget);
```

For this example, assume that data values of 1000 and 2000 are loaded into Budget -> Total Expenses for Colas and Root Beer, respectively. These values need to be allocated to each expense category, evenly spreading the values based on the non-missing children of Total Expenses from PY Actual. The allocated values need to be rounded to the nearest dollar.

This calculation script defines the allocation:

```
/* Allocate budgeted total expenses based on prior year */

/* Allocate budgeted total expenses based on prior year */

FIX("Total Expenses")
Budget = @ALLOCATE(Budget->"Total Expenses",
  @CHILDREN("Total Expenses"),"PY Actual",,
    spread,SKIPMISSING,roundAmt,0,errorsToHigh)
ENDFIX
```

This table shows the results:

		Budget	PY Actual
Colas	Marketing	334*	150
	Payroll	#MI	#MI
	Lease	333	200
	Misc	333	100
	Total Expenses	**1000**	**450**
Root Beer	Marketing	500	300
	Payroll	500	200
	Lease	500	200
	Misc	500	400
	Total Expenses	**2000**	**1100**

* Rounding errors are added to this value. See step 5 for more information.

Analytic Services cycles through the database, performing the following calculations:

1. Analytic Services fixes on the children of Total Expenses. Using a FIX statement with @ALLOCATE may improve calculation performance.
2. For Budget -> Colas -> Marketing, Analytic Services divides 1 by the count of non-missing values for each expense category in PY Actual -> Colas for each month. In this case, 1 is divided by 3, because there are 3 non-missing expense values for Budget -> Colas.

3. Analytic Services takes the value from step 2 (.333), multiplies it by the value for Budget -> Colas -> Total Expenses (1000), and then rounds to the nearest dollar (333). This value is placed in Budget -> Colas -> Marketing.
4. Analytic Services repeats steps 2–3 for each expense category for Budget -> Colas and then for Budget -> Root Beer.
5. As specified in the calculation script, the allocated values are rounded to the nearest whole dollar. Analytic Services makes a second pass through the block to make the sum of the rounded values equal to the allocation value (for example, 1000 for Budget -> Colas -> Total Expenses). In this example, there is a rounding error of 1 for Budget -> Colas -> Total Expenses, because the expense categories add up to 999, not 1000, which is the allocation value. Because all the allocated values are identical (333), the rounding error of 1 is added to the first value in the allocation range, Budget -> Colas -> Marketing (thus a value of 334).

Allocating Across Multiple Dimensions

The following example uses the @MDALLOCATE function to allocate a loaded value for budgeted total expenses across three dimensions. The budgeted total expenses are allocated based on the actual values of the prior year.

The following example is based on the Sample Basic database. Assume that you have made the following modifications:

- Added a child, PY Actual, to the Scenario dimension
- Copied data from Actual into PY Actual
- Cleared data from Budget

For this example, a value of 750 (for Budget -> Total Expenses -> Product -> East -> Jan) needs to be allocated to each expense category for the children of product 100 across the states in the East. The allocation uses values from PY Actual to determine the percentage share that each category should receive.

This calculation script defines the allocation:

```
/* Allocate budgeted total expenses based on prior year, across
3 dimensions */

SET UPDATECALC OFF;
FIX (East, “100”, “Total Expenses”)

BUDGET = @MDALLOCATE(750,3,@CHILDREN(“100”),@CHILDREN("Total
Expenses"),@CHILDREN(East),"PY Actual",,share);
ENDFIX
```

This table shows the values for PY Actual:

		Jan			
		PY Actual			
		Marketing	**Payroll**	**Misc**	**Total Expenses**
100-10	New York	94	51	0	145
	Massachusetts	23	31	1	55
	Florida	27	31	0	58
	Connecticut	40	31	0	71
	New Hampshire	15	31	1	47
100-20	New York	199	175	2	376
	Massachusetts	#MI	#MI	#MI	#MI
	Florida	#MI	#MI	#MI	#MI
	Connecticut	26	23	0	49
	New Hampshire	#MI	#MI	#MI	#MI
100-30	New York	#MI	#MI	#MI	#MI
	Massachusetts	26	23	0	49
	Florida	#MI	#MI	#MI	#MI
	Connecticut	#MI	#MI	#MI	#MI
	New Hampshire	#MI	#MI	#MI	#MI

		Jan			
		PY Actual			
		Marketing	Payroll	Misc	Total Expenses
100	New York	#MI	#MI	#MI	#MI
	Massachusetts	12	22	1	35
	Florida	12	22	1	35
	Connecticut	94	51	0	145
	New Hampshire	23	31	1	55
	East	237	220	3	460

Analytic Services cycles through the database, performing these calculations:

1. Analytic Services fixes on East, the children of 100, and Total Expenses. Using a FIX statement with @MDALLOCATE may improve calculation performance.
2. Before performing the allocation, Analytic Services needs to determine what share of 750 (the value to be allocated) each expense category should receive, for each product-state combination. To determine the share, Analytic Services uses the shares of each expense category from PY Actual. Starting with PY Actual -> 100-10 -> New York, Analytic Services divides the value for the first expense category, Marketing, by the value for PY Actual-> 100-10 -> East -> Total Expenses to calculate the percentage share of that category. For example, Analytic Services divides the value for PY Actual -> 100-10 -> New York -> Marketing (94) by the value for PY Actual -> 100-10 -> East -> Total Expenses (460), which yields a percentage share of approximately 20.4% for the Marketing category.
3. Analytic Services repeats step 2 for each expense category, for each product-state combination.
4. During the allocation, Analytic Services uses the percentage shares calculated in step 2 to step 3 to determine what share of 750 should be allocated to each child of Total Expenses from Budget, for each product-state combination. For example, for Marketing, Analytic Services uses the 20.4% figure calculated in step 2, takes 20.4% of 750 (approximately 153), and places the allocated value in Budget -> 100-10 -> New York -> Marketing (see the next table).

5. Analytic Services repeats step 4 for each expense category and for each product-state combination, using the percentage shares from PY Actual calculated in step 2 to step 3.
6. Analytic Services consolidates the expense categories to yield the values for Total Expenses.

This table shows the results of the allocation for Budget:

		Jan Budget			
		Marketing	**Payroll**	**Misc**	**Total Expenses**
100-10	New York	153.26	83.15	0	236.41
	Massachusetts	37.50	50.54	1.63	89.67
	Florida	44.02	50.54	0	94.56
	Connecticut	65.22	50.54	0	115.76
	New Hampshire	24.46	50.54	1.63	76.63
100-20	New York	#MI	#MI	#MI	#MI
	Massachusetts	#MI	#MI	#MI	#MI
	Florida	42.39	37.50	0	79.89
	Connecticut	#MI	#MI	#MI	#MI
	New Hampshire	#MI	#MI	#MI	#MI
100-30	New York	#MI	#MI	#MI	#MI
	Massachusetts	#MI	#MI	#MI	#MI
	Florida	#MI	#MI	#MI	#MI
	Connecticut	#MI	#MI	#MI	#MI
	New Hampshire	19.57	35.87	1.63	57.07

		Jan Budget			
		Marketing	**Payroll**	**Misc**	**Total Expenses**
100	New York	153.26	83.15	0	236.41
	Massachusetts	37.50	50.54	1.63	89.67
	Florida	86.41	88.04	0	174.46
	Connecticut	65.22	50.54	0	115.76
	New Hampshire	44.02	86.41	3.26	133.70
	East	386.41	358.70	4.89	750

Goal Seeking Using the LOOP Command

The following example is based on the Sample Basic database. However, the example assumes that no members are tagged as Dynamic Calc and that the Profit per Ounce member (under Ratios in the Scenario dimension) is not included in the calculation. For an explanation of how you calculate values dynamically and how you benefit from doing so, see Dynamic Calc members, in Chapter 25, "Dynamically Calculating Data Values."

You want to know what sales value you have to reach in order to obtain a certain profit on a specific product.

This example adjusts the Budget value of Sales to reach a goal of 15,000 Profit for Jan. The results are shown for product 100-10.

Figure 204: Measures Dimension from the Sample Basic Database

```
Measures Accounts (Label Only)
    Profit (+)
        Margin (+)
            Sales (+)
            COGS (-) (Expense Reporting)
        Total Expenses (-) (Expense Reporting)
            Marketing (+) (Expense Reporting)
            Payroll (+) (Expense Reporting)
            Misc (+) (Expense Reporting)
    Inventory (~) (Label Only)
    Ratios (~) (Label Only)
        Margin % (+) (Two Pass Calc) Margin % Sales;
        Profit % (~) (Two Pass Calc) Profit % Sales;
```

Assume that the data values before running the goal-seeking calculation script are as follows:

Product, Market, Budget	Jan
Profit	12,278.50
Margin	30,195.50
Sales	49,950.00
COGS	19,755.00
Total Expenses	17,917.00
Marketing	3,515.00
Payroll	14,402.00
Misc	0
Inventory	Label Only member
Ratios	Label Only member
Margin %	60.45
Profit %	24.58

This calculation script produces the goal-seeking results:

```
/* Declare the temporary variables and set their initial values*/

VAR

      Target = 15000,
      AcceptableErrorPercent = .001,
      AcceptableError,
      PriorVar,
      PriorTar,
      PctNewVarChange = .10,
      CurTarDiff,
      Slope,
      Quit = 0,
      DependencyCheck,
      NxtVar;

/*Declare a temporary array variable called Rollback and base it on the
Measures dimension */
ARRAY Rollback [Measures];
```

```
/* Fix on the appropriate member combinations and perform the goal-seeking
calculation*/
FIX(Budget, Jan, Product, Market)
      LOOP (35, Quit)
            Sales (Rollback = Budget;
            AcceptableError = Target * (AcceptableErrorPercent);
            PriorVar = Sales;
            PriorTar = Profit;
            Sales = Sales + PctNewVarChange * Sales;);
            CALC DIM(Measures);
            Sales (DependencyCheck = PriorVar - PriorTar;
            IF(DependencyCheck <> 0) CurTarDiff = Profit - Target;
                  IF(@ABS(CurTarDiff) > @ABS(AcceptableError))
                        Slope = (Profit - PriorTar) / (Sales - PriorVar);
                        NxtVar = Sales - (CurTarDiff / Slope);
                        PctNewVarChange = (NxtVar - Sales) / Sales;
                  ELSE
                        Quit = 1;
                  ENDIF;
            ELSE
                  Budget = Rollback;
                  Quit = 1;
            ENDIF;);
      ENDLOOP
      CALC DIM(Measures);
ENDFIX
```

Analytic Services performs the following calculations:

1. It declares the required temporary variables using the VAR command. Where appropriate, the initial values are set.
2. Analytic Services declares a one-dimensional array called Rollback. The size of Rollback is based on the number of members in the Measures dimension. Analytic Services uses Rollback to store the Budget values.
3. Analytic Services fixes on the Jan -> Budget values for all Product and Market members.
4. The LOOP command ensures that the commands between LOOP and ENDLOOP are cycled through 35 times *for each member combination*. However, if the Quit variable is set to 1, then the LOOP is broken and the calculation continues after the ENDLOOP command.

5. Analytic Services cycles through the member combinations, performing the following calculations:

   a. Analytic Services places the Budget -> Sales value in the Rollback temporary array variable.

   b. It calculates the acceptable error. It multiplies the Target value (15000) by the AcceptableErrorPercent value (0.001) and places the result in the AcceptableError variable.

   c. It retains the current Sales value. It places the Sales value for the current member combination in the PriorVar temporary variable.

   d. It retains the current Profit value. It places the Profit value for the current member combination in the PriorTar temporary variable.

   e. It calculates a new Sales value. It multiplies the PctNewVarChange value (0.1) by the current Sales value, adds the current Sales value, and places the result in Sales.

   f. Analytic Services calculates and consolidates the Measures dimension.

   g. It subtracts the PriorTar value from the PriorVar value and places the result in the DependencyCheck temporary variable.

   h. The IF command checks that DependencyCheck is not 0 (zero).

      - If DependencyCheck is not 0, then Analytic Services subtracts the Target value (15000) from the current Profit and places the result in the CurTarDiff temporary variable.

        The IF command checks to see if the absolute value (irrespective of the + or – sign) of CurTarDiff is greater than the absolute value of the acceptable error (AcceptableError). If it is, Analytic Services calculates the Slope, NxtVar, and PctNewVarChange temporary variables.

        If it is not greater than AcceptableError, Analytic Services breaks the LOOP command by setting the value of Quit to 1. The calculation continues after the ENDLOOP command.

      - If DependencyCheck is 0, Analytic Services places the value in the Rollback array into Budget. Analytic Services breaks the LOOP command by setting the value of Quit to 1. The calculation continues after the ENDLOOP command.

6. Analytic Services calculates and consolidates the Measures dimension.

The results are shown in this table:

Product, Market, Budget	Jan
Profit	15,000.00
Margin	32,917.00
Sales	52,671.50
COGS	19,755.00
Total Expenses	17,917.00
Marketing	3,515.00
Payroll	14,402.00
Misc	0
Inventory	Label Only member
Ratios	Label Only member
Margin %	28.47839913
Profit %	62.49489762

Forecasting Future Values

The following example uses the @TREND function to forecast sales data for June through December, assuming that data currently exists only up to May. Using the linear regression forecasting method, this example produces a trend, or line, that starts with the known data values from selected previous months and continues with forecasted values based on the known values. In addition, this example demonstrates how to check the results of the trend for "goodness of fit" to the known data values.

The following example is based on the Sample Basic database. Assume that the Measures dimension contains an additional child, ErrorLR. The goodness-of-fit results are placed in this member. This calculation script defines the forecasting:

```
Sales
(@TREND(@LIST(Jan,Mar,Apr),@LIST(1,3,4),,
  @RANGE(ErrorLR,@LIST(Jan,Mar,Apr)),
    @LIST(6,7,8,9,10,11,12),
      Jun:Dec,LR););
```

This table explains each parameter:

Parameter	Description
@LIST(Jan,Mar,Apr)	Represents the *Ylist*, or the members that contain the known data values. The @LIST function is needed to group the three members as a comma-delimited list and to keep the list separate from other parameters.
@LIST(1,3,4)	Represents the *Xlist*, or the underlying variable values. Since Feb and May are skipped, Analytic Services needs to number the *Ylist* values accordingly (1,3,4).
,	The extra comma after the *Xlist* parameter indicates that a parameter has been skipped, in this case, the *weightList* parameter. The default weight of 1 is used for this example.
@RANGE(ErrorLR, @LIST(Jan,Mar,Apr)	Represents the *errorList*, or the member list where results of the goodness of fit of the trend line to *Ylist* are placed. The values placed in *errorList* are the differences between the data points in *Ylist* and the data points on the trend line produced. The @RANGE function combines the ErrorLR member with *Ylist* (Jan, Mar, Apr) to produce a member list.
@LIST(6,7,8,9,10,11,12)	Represents the *XforecastList*, or the underlying variable values for which the forecast is sought. This example forecasts values consecutively for Jun through Dec, so the values are simply 6,7,8,9,10,11,12.

Parameter	Description
Jun:Dec	Represents the *YforecastList*, or the member list into which the forecast values are placed. In this example, values are forecast for Jun through Dec based on the values for Jan, Mar, and Apr.
LR	Specifies the Linear Regression method.

This table shows the results of the calculation script:

	100 West Actual	
	Sales	ErrorLR
Jan	2339	4.57
Feb	2298	#MI
Mar	2313	-13.71
Apr	2332	9.14
May	2351	#MI
Jun	2315.14	#MI
Jul	2311.29	#MI
Aug	2307.49	#MI
Sep	2303.57	#MI
Oct	2299.71	#MI
Nov	2295.86	#MI
Dec	2292	#MI

Analytic Services cycles through the database, performing the following calculations:

1. Analytic Services finds the known data values on which to base the trend (Sales for Jan, Mar, Apr), as specified for the *Ylist* and *Xlist* parameters in the calculation script.
2. Analytic Services calculates the trend line using Linear Regression and places the results in Sales for Jun through Dec, as specified for the *YforecastList* parameter in the calculation script.
3. Analytic Services calculates the goodness of fit of the trend line to the data values for Jan, Mar, and Apr and places the results in ErrorLR for those months. For example, the value in ErrorLR for Jan (4.57) means that after Analytic Services calculates the trend line, the difference between the Sales value for Jan (2339) and the Jan value on the trend line is 4.57. The ErrorLR values for Feb and May are #MISSING since these months were not part of *Ylist*.

Chapter 29

Developing Custom-Defined Calculation Macros

This chapter explains how to develop custom-defined macros and how to use them in calculation scripts and formulas. *Custom-defined macros* are written with calculator functions and special macro functions. Macros enable you to combine multiple calculation functions into a single function.

For more details about the macro language syntax and rules, and examples of the use of the macro language, see the *Technical Reference*.

This chapter includes the following sections:

Note: The information in this chapter is not relevant to aggregate storage databases.

Understanding Custom-Defined Macros

Custom-defined macros use an internal macro language that enables you to combine calculation functions and operate on multiple input parameters.

When developing and testing custom-defined macros, make sure to create and test new macros locally within a test application. You should register custom-defined macros globally only after you have tested them and are ready to use them as part of a production environment.

Analytic Services requires that you have a security level of database designer or higher to create and manage custom-defined macros.

Viewing Custom-Defined Macros

View a custom-defined macro to determine whether a macro has been successfully created or whether a custom-defined macro is local or global.

➤ To view a custom-defined macro, use either of the following methods:

Tool	Topic	Location
Administration Services	Viewing Custom-Defined Macros	*Essbase Administration Services Online Help*
MaxL	**display macro**	*Technical Reference*

Creating Custom-Defined Macros

When you create a custom-defined macro, Analytic Services records the macro definition and stores it for future use. Create the macro once, and then you can use it in formulas and calculation scripts until the macro is updated or removed from the catalog of macros.

Use these sections to understand more about creating custom-defined macros and to find instructions for creating them:

- “Understanding Scope” on page 633
- “Naming Custom-Defined Macros” on page 633
- “Creating Macros” on page 634
- “Refreshing the Catalog of Custom-Defined Macros” on page 634

Understanding Scope

You can create custom-defined macros locally or globally. When you create a local custom-defined macro, the macro is only available in the application in which it was created. When you create a global custom-defined macro, the macro is available to all applications on the server where it was created.

When you create a global custom-defined macro, all Analytic Services applications can use it. Be sure you test custom-defined macros in a single application (and create them only in that application) before making them global macros.

CAUTION: When testing macros, do not create the macros as global (using a macro name without the *AppName.* prefix), because global macros are difficult to update. For information about the methods for updating custom-defined macros, see "Updating Custom-Defined Macros" on page 636.

For rules about naming custom-defined macros, see "Naming Custom-Defined Macros" on page 633.

Naming Custom-Defined Macros

Remember these requirements when you create macro names:

- The names of custom-defined macro must start with an "@" symbol; for example, @MYMACRO. The rest of a macro name can contain letters, numbers, and the following symbols: @, #, $, and _. Macro names can not contain spaces.
- The names of custom-defined macros which are only called by other macros should start with "@_" to distinguish them from general use macros and functions.
- Macros must have unique names. Macro names must be different from each other, from the names of custom-defined functions, and from the names of existing calculation functions. If an application contains a local macro that has the same name as a global macro, the local macro takes precedence and is used for calculation.

Creating Macros

➤ To create a custom-defined macro, use either of the following methods:

Tool	Topic	Location
Administration Services	Creating Custom-Defined Macros	*Essbase Administration Services Online Help*
MaxL	**create macro**	*Technical Reference*

Be sure to add the application name plus a period (.) as a prefix before the name of the local macro. In this example, Sample is the prefix for the local macro name. This prefix assigns the macro to an application, so the macro is only available within that application.

For example, use the following MaxL statement to create a local macro named @COUNTRANGE used only in the Sample application:

```
create macro Sample.'@COUNTRANGE'(Any) AS
'@COUNT(SKIPMISSING, @RANGE(@@S))'
spec '@COUNTRANGE(MemberRange)'
comment 'counts all non-missing values';
```

For example, use the following MaxL statement to create a global macro named @COUNTRANGE:

```
create macro'@COUNTRANGE'(Any) AS
'@COUNT(SKIPMISSING, @RANGE(@@S))'
spec '@COUNTRANGE(MemberRange)'
comment 'counts all non-missing values';
```

Refreshing the Catalog of Custom-Defined Macros

➤ To refresh the catalog of custom-defined macros for all applications on a server, restart the server.

➤ To refresh the catalog of custom-defined macros for a single application, use the **refresh custom definitions** MaxL statement.

For example, use the following MaxL statement to refresh the catalog of custom-defined macros for the Sample application:

```
refresh custom definition on application sample;
```

Using Custom-Defined Macros

After creating custom-defined macros, you can use them like native calculation commands. Local macros—created using the *AppName.* prefix on the macro name—are only available for use in calculation scripts or formulas within the application in which they were created. Global macros—created without the *AppName.* prefix—are available to all calculation scripts and formulas on the server where they were created.

For a comprehensive discussion of creating custom-defined macros, see "Creating Custom-Defined Macros" on page 632.

➤ To use a custom-defined macro follow these steps:

1. Create a new calculation script or formula, or open an existing calculation script or formula.
   - If the custom-defined macro was registered locally—within a specific application—you must use a calculation script or formula within that application.
   - If the custom-defined macro was registered globally, you can use any calculation script or formula on any application on the server.
2. Add the custom-defined macro to a new or existing calculation script or formula. To use the custom-defined macro shown earlier in this chapter, type the following calculation script:

   ```
   CountMbr = @COUNTRANGE(Sales, Jan:Dec);
   ```

   Use this calculation script with the Sample Basic database, or replace "Sales, Jan:Dec" with a range of members in a test database.
3. Save the calculation script or formula, and then run it as usual.

For information about creating and running calculation scripts, see Chapter 27, "Developing Calculation Scripts." For information about creating and running formulas, see Chapter 22, "Developing Formulas."

Updating Custom-Defined Macros

When you update a custom-defined macro, you must determine whether the macro is registered locally or globally. Local custom-defined macros are created using an *AppName.* prefix in the macro name and can be used only within the application where they were created. For a review of the methods used to determine whether a custom-defined macro is local or global, see "Viewing Custom-Defined Macros" on page 632. For a review of the methods used to update the catalog of macros after you change a custom-defined macro, see "Refreshing the Catalog of Custom-Defined Macros" on page 634.

➤ To update a custom-defined macro, use either of the following methods:

Tool	Topic	Location
Administration Services	Editing Custom-Defined Macros	*Essbase Administration Services Online Help*
MaxL	**create or replace macro**	*Technical Reference*

For example, use the following MaxL statement to change the local macro @COUNTRANGE which is used only in the Sample application:

```
create or replace macro Sample.'@COUNTRANGE'(Any)
as '@COUNT(SKIPMISSING, @RANGE(@@S))';
```

For example, use the following MaxL statement to change the global macro @COUNTRANGE:

```
create or replace macro '@COUNTRANGE'(Any)
as '@COUNT(SKIPMISSING, @RANGE(@@S))';
```

Copying Custom-Defined Macros

You can copy custom-defined macros to any Analytic Server and application to which you have appropriate access.

➤ To copy a custom-defined macro, use either of the following methods:

Tool	Topic	Location
Administration Services	Copying Custom-Defined Macros	*Essbase Administration Services Online Help*
MaxL	**create macro**	*Technical Reference*

Removing Custom-Defined Macros

When removing a custom-defined macro, you must first determine whether the macro is registered locally or globally. The procedure for removing global custom-defined macros is more complex than that for removing local custom-defined macros and should only be performed by a database administrator. For a review of methods used to determine whether a custom-defined macro is local or global, see "Viewing Custom-Defined Macros" on page 632.

Before removing custom-defined macros, verify that no calculation scripts or formulas are using them. Global custom-defined macros can be used in calculation scripts and formulas across a server, so verify that no calculation scripts or formulas on the server are using a global custom-defined macro before removing it.

For a review of methods used to update the catalog of macros after you remove a custom-defined macro, see "Refreshing the Catalog of Custom-Defined Macros" on page 634.

➤ To remove a custom-defined macro, use either of the following methods:

Tool	Topic	Location
Administration Services	Deleting Custom-Defined Macros	*Essbase Administration Services Online Help*
MaxL	**drop macro**	*Technical Reference*

For example, use the following MaxL statement to remove the local macro @COUNTRANGE which is used only in the Sample application:

```
drop macro Sample.'@COUNTRANGE';
```

For example, use the following MaxL statement to remove the global macro @COUNTRANGE:

```
drop macro '@COUNTRANGE';
```

Chapter 30

Developing Custom-Defined Calculation Functions

This chapter explains how to develop custom-defined functions and use them in Analytic Services formulas and calculation scripts. Custom-defined functions are written in the Java™ programming language and enable you to create calculation functions not otherwise supported by the Analytic Services calculation scripting language.

Analytic Services does not provide tools for creating Java classes and archives. This chapter assumes that you have a compatible version of the Java Development Kit (JDK) and a text editor installed on the computer you use to develop custom-defined functions. For information on compatible versions of Java, see the *Essbase Analytic Services Installation Guide*.

For additional examples of custom-defined functions, see the *Technical Reference*.

Use these sections to create and use custom-defined functions:

Note: The information in this chapter is not relevant to aggregate storage databases.

Viewing Custom-Defined Functions

You can view custom-defined functions to determine whether a function has been registered successfully and whether it is registered locally or globally. No custom-defined functions are displayed until they have been created and registered. Analytic Services does not supply sample custom-defined functions.

➤ To view a custom-defined function, use either of the following methods:

Tool	Topic	Location
Administration Services	Viewing Custom-Defined Functions	*Essbase Administration Services Online Help*
MaxL	**display function**	*Technical Reference*

For example, use the following MaxL statement to view the custom-defined functions in the Sample application and any registered global functions:

```
display function Sample;
```

The **display function** statement lists global functions without an application name to indicate that they are global. If the application contains a function with the same name as a global function, only the local function is listed.

Creating Custom-Defined Functions

There are several steps required to create a custom-defined function:

1. Learn the requirements for custom-defined functions.

   See the following sections for more information:

   - “Understanding Java Requirements for Custom-Defined Functions” on page 641
   - “Understanding Method Requirements for Custom-Defined Functions” on page 642
   - “Understanding Security and Custom-Defined Functions” on page 643
   - “Understanding Scope and Custom-Defined Functions” on page 643
   - “Naming Custom-Defined Functions” on page 643

2. Write a public Java class that contains at least one public, static method to be used as a custom-defined function.

   For instructions, see "Creating and Compiling the Java Class" on page 644.

3. Install the Java class created in step 2.

   For instructions, see "Installing Java Classes on Analytic Server" on page 645.

4. Register the custom-defined function as a local or global function.

   For instructions, see "Registering Custom-Defined Functions" on page 646.

Understanding Java Requirements for Custom-Defined Functions

The basis of a custom-defined function is a Java class and method created by a database administrator or Java programmer to perform a particular type of calculation. Creating and testing these Java classes and methods is the first step toward creating a custom-defined function.

You can create more than one method in a class for use as a custom-defined function. In general, it is recommended that you create all the methods you want to use as custom-defined functions in a single class. However, if you want to add new custom-defined functions that are not going to be used across all applications on the Analytic Server, create them in a new class and add them to the Analytic Server in a separate `.jar` file.

When creating multiple Java classes that contain methods for use as custom-defined functions, verify that each class name is unique. Duplicate class names cause methods in the duplicate class not to be recognized, and you cannot register those methods as custom-defined functions.

After creating the Java classes and methods for custom-defined functions, test them using test programs in Java. When you are satisfied with the output of the methods, install them on Analytic Server and register them in a single test application. Do not register functions globally for testing, because registering functions globally makes it more difficult to update them if you find problems.

Understanding Method Requirements for Custom-Defined Functions

Methods in custom-defined functions can have any combination of the following supported data types as input parameters:

Table 27: Data Types Allowed As Input Parameters

Type	Type
boolean	byte
char	java.lang.String
short, int, long	com.hyperion.essbase.calculator.CalcBoolean
float, double	arrays of any of these types

CalcBoolean is an Analytic Services-specific data type that can include three values—TRUE, FALSE, and #MISSING. For more information about the other listed data types, see the documentation for the Java Development Kit.

The method return data type can be void or any of the preceding data types. Returned data types are converted to Analytic Services-specific data types. Strings are mapped to a string type. Boolean values are mapped to the CalcBoolean data type. All other values are mapped to a double type.

Note: Double variables returned with infinite or Not-a-Number values are not supported by Analytic Services. If these values are returned from a Java program, they may not be recorded or displayed correctly in Analytic Services. Double variables should be checked for infinite or Not-a-Number values and set to finite values before being returned to Analytic Services. For more information, see the entry for the class, Double, in the documentation for the Java Development Kit.

For additional examples of Java custom-defined functions, see the MaxL statement **create function** in the *Technical Reference*.

Understanding Security and Custom-Defined Functions

Analytic Services requires that you have these security permissions:

- To create, delete, and manage local (application-wide) custom-defined functions, you must have security permission of application designer or higher.
- To create, delete, and manage global (server-wide) custom-defined functions, you must have security permission of supervisor.

Understanding Scope and Custom-Defined Functions

Custom-defined functions are registered as either local (application-wide) or global (Analytic Server-wide) in scope. When you register a local custom-defined function, it is available only in the application in which it was registered. When you register a global custom-defined function, it is available to all applications on the Analytic Server on which it was registered.

When developing and testing custom-defined functions, make sure to register and test new functions locally within a test application. You should never register custom-defined functions globally until you have thoroughly tested them and are ready to use them as part of a production environment.

Naming Custom-Defined Functions

When you register a custom-defined function in Analytic Services, you give the function a name. This name is used in calculation scripts and formulas and is distinct from the Java class and method name used by the function.

Remember these requirements when you create function names:

- Custom-defined function names must start with an @ symbol; for example, @MYFUNCTION. The rest of a function name can contain letters, numbers, and the following symbols: @, #, $, and _. Function names cannot contain spaces.
- The names of custom-defined functions which are called only by custom-defined macros should start with "@_" to distinguish them from general use functions and macros.

- Custom-defined functions must have unique names. Function names must be different from each other, from the names of custom-defined macros, and from the names of existing calculation functions.
- If an Analytic Services application contains a local function that has the same name as a global function, the local function is used for calculation.

For information about and instructions for registering custom-defined functions, see "Registering Custom-Defined Functions" on page 646.

Creating and Compiling the Java Class

➤ To create a Java class for a custom-defined function, use this procedure:

1. Start a text editor and create a Java class. For example, open a text editor and create this class:

```
public class CalcFunc {
  public static double sum (double[] data) {
    int i, n = data.length;
    double sum = 0.0d;
    for (i=0; i<n; i++) {
      double d = data [i];
      sum = sum + d;
    }
    return sum;
  }
}
```

2. Save the Java class as a .java file; for example, `CalcFunc.java`.
3. Compile the Java class using the JDK javac tool. At the operating system command prompt, move to the directory containing the class you want to compile and use the javac tool. For example, to compile the `CalcFunc.java` class, type this command:

```
>javac CalcFunc.java
```

4. Resolve any compiling errors and repeat steps 2 and 3 until the compiler creates a new `.class` file; for example, `CalcFunc.class`.

Installing Java Classes on Analytic Server

When you install Java classes on Analytic Server, you must first compile the classes into a Java Archive (`.jar`) file, and then copy the `.jar` file to a specific location on the computer running Analytic Server.

Note: You must either stop and restart Analytic Services applications or stop and restart Analytic Server when you install new `.jar` files.

➤ To create a `.jar` file from a `.class` file and install it on an Analytic Server:

1. At the operating system command prompt, move to the directory containing the class you want to add to a `.jar` file and use the JDK jar tool. To add `CalcFunc.class` to a `.jar` file, type this command:

   ```
   >jar cf CalcFunc.jar CalcFunc.class
   ```

2. Copy the `.jar` file to one of the following locations on the computer running Analytic Server:

   - The *ARBORPATH*`\java\udf\` directory. This location is recommended for `.jar` files containing global custom-defined functions. If this directory does not exist, create the directory.
   - The *ARBORPATH*`\app\`*AppName*`\udf\` directory where *AppName* is the name of the application where the local custom-defined function will be used. If this directory does not exist, create the directory. Use this location if you want the code in the `.jar` file to only be used with a specific application.

   If the `.jar` file is placed in another location, you must modify the CLASSPATH variable to include the full path and file name for the `.jar` file.

3. If the functions will only be used by specific applications, restart those applications in Analytic Services. Otherwise, restart Analytic Server.

Registering Custom-Defined Functions

After you have compiled the Java classes for custom-defined functions into `.jar` files and installed the `.jar` files on Analytic Server, you must register the custom-defined functions before you can use them in calculation scripts and formulas. For rules about naming custom-defined functions, see "Naming Custom-Defined Functions" on page 643.

When you register a global custom-defined function, all Analytic Services applications on the Analytic Server can use it. Be sure you test custom-defined functions in a single application (and register them only in that application) before making them global functions.

Use the same process for updating the catalog of functions as you do for updating the catalog of macros. After you register a custom-defined function, see "Refreshing the Catalog of Custom-Defined Macros" on page 634.

CAUTION: Do not register global functions for testing, because registering them globally makes it more difficult to change them if you find problems.

➤ To register a custom-defined function, use either of the following methods:

Tool	Topic	Location
Administration Services	Creating Custom-Defined Functions	*Essbase Administration Services Online Help*
MaxL	**create function**	*Technical Reference*

To register a custom-defined function with local scope, include the application name as a prefix. For example, use the following MaxL statement to register the local custom-defined function in the class CalcFunc from "Creating and Compiling the Java Class" on page 644 to be used within the Sample application:

```
create function Sample.'@JSUM'
as 'CalcFunc.sum'
spec '@JSUM(memberRange)'
comment 'adds list of input members';
```

To register a custom-defined function with global scope, do not include the application name as a prefix. For example, use the following MaxL statement to register as a global function the custom-defined function in the class CalcFunc from "Creating and Compiling the Java Class" on page 644:

```
create function '@JSUM'
as 'CalcFunc.sum';
spec '@JSUM(memberRange)'
comment 'adds list of input members';
```

The *AppName*. prefix is not included in the name of the function. The lack of a prefix makes a function global.

Note: Specifying input parameters for the Java method is optional. If you do not specify input parameters, Analytic Services reads them from the method definition in the Java code. However, if you are registering two or more custom-defined functions with the same method name but with different parameter sets, you must register each version of the function separately, specifying the parameters for each version of the function.

Using Registered Custom-Defined Functions

After registering custom-defined functions, you can use them like native Analytic Services calculation commands. Functions you registered locally—using the *AppName*. prefix on the function name—are only available for use in calculation scripts or formulas within the application in which they were registered. If you registered the custom-defined functions globally, then the functions are available to all calculation scripts and formulas on the Analytic Server where the functions are registered.

For information and instructions for registering custom-defined functions, see "Registering Custom-Defined Functions" on page 646.

➤ To use a registered custom-defined function:

1. Create a new calculation script or formula, or open an existing calculation script or formula.
   - If the custom-defined function was registered locally—within a specific application—then you must use a calculation script or formula within that application.
   - If the custom-defined function was registered globally, then you can use any calculation script or formula on Analytic Server.

2. Add the custom-defined function to a new or existing calculation script or formula. To use the custom-defined function shown earlier in this chapter, use this calculation script:

   ```
   "New York" = @JSUM(@LIST(2.3, 4.5, 6.6, 1000.34));
   ```

   Use this calculation script with the Sample Basic sample database, or replace "New York" with the name of a member in a test database.

3. Save the calculation script or formula, and then run it as usual.

For a comprehensive discussion of creating and running calculation scripts, see Chapter 27, "Developing Calculation Scripts." For a comprehensive discussion of creating and running formulas, see Chapter 22, "Developing Formulas."

Updating Custom-Defined Functions

When you update a custom-defined function, there are two major issues to consider:

- Is the function registered locally or globally?
- Have the class name, method name, or input parameters changed in the Java code for the custom-defined function?

The answers to these questions determine the procedure for updating the custom-defined function.

For a review of methods to determine whether a custom-defined function is local or global, see "Viewing Custom-Defined Functions" on page 640.

To update a custom-defined function, in most cases, you must replace the `.jar` file that contains the code for the function, and then re-register the function. However, if the signature of the custom-defined function—the Java class name, method name and input parameters—have not changed and the function has only one set of input parameters (it is not an overloaded method), you can simply replace the `.jar` file that contains the function. In either situation, you must stop and restart the Analytic Services application or Analytic Server where the custom-defined function is registered, depending on whether it is a local or global function.

Note: The following procedure is effective only if the signature of the custom-defined function—the Java class name, method name, and input parameters for the custom-defined function—has not changed.

➤ To update a custom-defined function whose signature has not changed:

1. Make the changes to the Java class for the custom-defined function and use Java test programs to test its output.
2. Compile the Java classes and encapsulate them in a `.jar` file, using the same name as the previous `.jar` file. Make sure to include any other classes and methods for custom-defined functions that were included in the previous `.jar` file.
3. Perform one of the following steps:
   - If the function is registered locally, shut down the application in which it was registered.
   - If the function is registered globally, shut down all running applications.
4. Copy the new `.jar` file to Analytic Server over the existing `.jar` file with the same name.
5. Restart the applications you shut down in step 3.

Updating Local Custom-Defined Functions

Local custom-defined functions are registered with an *AppName*. prefix on the function name and can be used only within the application where they were registered. To update a local custom-defined function, you must stop and restart the Analytic Services application where the function is registered.

➤ To update a local custom-defined function:

1. Make the changes to the Java class for the custom-defined function and use Java test programs to test its output.
2. Compile the Java classes and encapsulate them in a `.jar` file, using the same name as the previous `.jar` file. Make sure to include any other classes and methods for custom-defined functions that were included in the previous `.jar` file.
3. Perform one of the following steps:
   - If the `.jar` file contains local custom-defined functions only, shut down any Analytic Services applications that use these functions.
   - If the `.jar` file contains any global custom-defined functions, shut down all Analytic Services applications.

- If you are unsure as to which Analytic Services applications use which functions in the `.jar` file you are replacing, shut down all Analytic Services applications.

4. Copy the new `.jar` file to Analytic Server over the existing `.jar` file with the same name.
5. Use MaxL or Essbase Administration Services to replace the custom-defined function.

   - In MaxL, use the **create or replace function** statement to replace the custom-defined function; for example, type this statement:

     ```
     create or replace function sample.'@JSUM'
     as 'CalcFunc.sum';
     ```

   - In Essbase Administration Services, see "Editing Custom-Defined Functions" in *Essbase Administration Services Online Help*.

Updating Global Custom-Defined Functions

Global custom-defined functions are registered without an *AppName.* prefix on the function name and are available in any application running on the Analytic Server where they are registered. To update a global custom-defined function, you must stop and restart all applications on Analytic Server.

Global custom-defined functions can be used in calculation scripts and formulas across Analytic Server, so you should verify that no calculation scripts or formulas are using a global custom-defined function before updating it.

CAUTION: Only a database administrator should update global custom-defined functions.

➤ To update a global custom-defined function:

1. Make the changes to the Java class for the custom-defined function and use Java test programs to test its output.
2. Compile the Java classes and encapsulate them in a `.jar` file, using the same name as the previous `.jar` file. Make sure to include any other classes and methods for custom-defined functions that were included in the previous `.jar` file.

3. Shut down all running Analytic Services applications.
4. Copy the new `.jar` file to Analytic Server over the existing `.jar` file with the same name.
5. Use MaxL or Essbase Administration Services to replace the custom-defined function.
   - In MaxL, use the **create or replace function** statement to replace the custom-defined function; for example, type this statement:

     ```
     create or replace function '@JSUM'
     as 'CalcFunc.sum';
     ```

   - In Essbase Administration Services, see "Editing Custom-Defined Functions" in *Essbase Administration Services Online Help*.

Removing Custom-Defined Functions

When removing a custom-defined function, you must first determine whether the function is registered locally or globally to identify the security permissions required:

- To remove a custom-defined function with global scope, you must have supervisor permission.
- To remove a custom-defined function with local scope, you must have application designer permission for the application, or any wider permission.

Before removing custom-defined functions, you should verify that no calculation scripts or formulas are using them. Global custom-defined functions can be used in calculation scripts and formulas across Analytic Server, so you must verify that no calculation scripts or formulas on Analytic Server are using a global custom-defined function before removing it.

For a review of methods to determine whether a custom-defined function is local or global, see "Viewing Custom-Defined Functions" on page 640.

Removing Local Custom-Defined Functions

Local custom-defined functions are registered with an *AppName.* prefix on the function name and can only be used within the application where they are registered.

When you remove a local custom-defined function, be sure to stop and restart the Analytic Services application where the function is registered to update the catalog.

➤ To remove a custom-defined function, use either of the following methods:

Tool	Topic	Location
Administration Services	Deleting Custom-Defined Functions	*Essbase Administration Services Online Help*
MaxL	**drop function**	*Technical Reference*

For example, use the following MaxL statement to remove the @JSUM function from the Sample application:

```
drop function Sample.'@JSUM';
```

Removing Global Custom-Defined Functions

Global custom-defined functions are registered without an *AppName.* prefix on the function name and are available in any application running on Analytic Server where they are registered.

Be sure to stop and restart all running Analytic Services applications when you remove a global custom-defined function.

Global custom-defined functions can be used in calculation scripts and formulas across Analytic Server, so you should verify that no calculation scripts or formulas are using a global custom-defined function before removing it.

CAUTION: Only a database administrator with supervisor permission can remove global custom-defined functions. Removal of global custom-defined functions should only be performed when no users are accessing Analytic Services databases and no calculation routines are being performed.

➤ To remove a global custom-defined function, use either of the following methods:

Tool	Topic	Location
Administration Services	Deleting Custom-Defined Functions	*Essbase Administration Services Online Help*
MaxL	**drop function**	*Technical Reference*

For example, use the following MaxL statement to remove the global @JSUM function:

```
drop function '@JSUM';
```

Copying Custom-Defined Functions

You can copy custom-defined functions to any Analytic Server and application to which you have appropriate access.

To copy a custom-defined macro, use either of the following methods:

Tool	Topic	Location
Administration Services	Copying Custom-Defined Functions	*Essbase Administration Services Online Help*
MaxL	**create function**	*Technical Reference*

Considering How Custom-Defined Functions Affect Performance and Memory

The ability to create and run custom-defined functions is provided as an extension to the Analytic Services calculator framework. When you use custom-defined functions, consider how their use affects memory resources and calculator performance.

Performance Considerations

Because custom-defined functions are implemented as an extension of the Analytic Services calculator framework, you can expect custom-defined functions to operate less efficiently than functions that are native to the Analytic Services calculator framework. In tests using a simple addition function running in the Java Runtime Environment 1.3 on the Windows NT 4.0 platform, the Java function ran 1.8 times (about 80%) slower than a similar addition function performed by native Analytic Services calculation commands.

To optimize performance, limit use of custom-defined functions to calculations that you cannot perform with native Analytic Services calculation commands, particularly in applications where calculation speed is a critical consideration.

Memory Considerations

Use of the Java Virtual Machine (JVM) and Java API for XML Parsing has an initial effect on the memory required to run Analytic Services. The memory required to run these additional components is documented in the memory requirements for Analytic Services. For more information about memory requirements, see the *Essbase Analytic Services Installation Guide*.

Beyond these start-up memory requirements, the Java programs you develop for custom-defined functions sometimes require additional memory. When started, the JVM for Win32 operating systems immediately allocates 2 MB of memory for programs. This allocation is increased according to the requirements of the programs that are then run by the JVM. The default upper limit of memory allocation for the JVM on Win32 operating systems is 64 MB. If the execution of a Java program exceeds the default upper limit of memory allocation for the JVM, the JVM generates an error. For more information about JVM memory management and memory allocation details for other operating systems, see the Java Development Kit documentation.

Considering the default memory requirements of the JVM and the limitations of the hardware on which you run servers, carefully monitor your use of memory. In particular, developers of custom-defined functions should be careful not to exceed memory limits of the JVM when creating large objects within custom-defined functions.

Part V

Retrieving Data

Part V describes some of the methods for retrieving data from Analytic Services databases, including how to develop report scripts, export data to other programs, and using data mining tools to more deeply analyze the data.

- Chapter 31, "Understanding Report Script Basics," introduces you to the basic concepts behind reports and describes how to create them using the Report Writer.
- Chapter 32, "Developing Report Scripts," describes how to create complex report scripts, including page layout and formatting information, how to select and sort members, how to restrict and order data values, how to convert data to a different currency, and how to generate reports using the C, Visual Basic, and Grid APIs.
- Chapter 33, "Mining an Analytic Services Database" describes how to use data-mining algorithms to find hidden relationships in large amounts of data.
- Chapter 34, "Copying Data Subsets and Exporting Data to Other Programs," describes how to move data from Analytic Services databases to other programs by extracting an output file of the data to move using the Report Writer.

Note: For information about optimizing report generation and data retrieval, see Chapter 56, "Optimizing Reports and Other Types of Retrieval." For extensive examples of report scripts, see the *Technical Reference*.

Chapter 31

Understanding Report Script Basics

An Analytic Services report enables you retrieve formatted summaries from an Analytic Services database. This chapter provides fundamental information about working with report scripts created by Report Writer. This chapter contains the following sections:

For a comprehensive discussion of creating complex report scripts, see Chapter 32, "Developing Report Scripts."

Creating a Simple Report Script

When you combine report commands that include page, row, and column dimension declarations with selected members, you have all the elements of a simple report script.

The following step-by-step example of a report script specifies these elements, dimensions, and member selection commands. It includes comments, which document the behavior of the script, and the **!** output command. This example is based on the Sample Basic database, which is supplied with the Analytic Server installation.

1. Create a report script.

   For more information, see "Creating Scripts" in the *Essbase Administration Services Online Help*.

2. Type the following information in the report, with the exception of the commented (//) lines, which are for your reference:

```
// This is a simple report script example
// Define the dimensions to list on the current page, as below
<PAGE (Market, Measures)

// Define the dimensions to list across the page, as below
<COLUMN (Year, Scenario)

// Define the dimensions to list down the page, as below
<ROW (Product)

// Select the members to include in the report
Sales
<ICHILDREN Market
Qtr1 Qtr2
Actual Budget Variance
<ICHILDREN Product

// Finish with a bang
    !
```

3. Save the report script.

   For more information, see "Saving Scripts" in the *Essbase Administration Services Online Help*.

4. Execute the report script.

   For information, see "Executing Report Scripts" in the *Essbase Administration Services Online Help*.

When you execute this report against the Sample Basic database, the script produces the following report:

East Sales

	Qtr1			Qtr2		
	Actual	Budget	Variance	Actual	Budget	Variance
100	9,211	6,500	2,711	10,069	6,900	3,169
200	6,542	3,700	2,842	6,697	3,700	2,997
300	6,483	4,500	1,983	6,956	5,200	1,756
400	4,725	2,800	1,925	4,956	3,200	1,756
Product	26,961	17,500	9,461	28,678	19,000	9,678

West Sales

	Qtr1			Qtr2		
	Actual	Budget	Variance	Actual	Budget	Variance
100	7,660	5,900	1,760	7,942	6,500	1,442
200	8,278	6,100	2,178	8,524	6,200	2,324
300	8,599	6,800	1,799	9,583	7,600	1,983
400	8,403	5,200	3,203	8,888	6,300	2,588
Product	32,940	24,000	8,940	34,937	26,600	8,337

South Sales

	Qtr1			Qtr2		
	Actual	Budget	Variance	Actual	Budget	Variance
100	5,940	4,100	1,840	6,294	4,900	1,394
200	5,354	3,400	1,954	5,535	4,000	1,535
300	4,639	4,000	639	4,570	3,800	770
400	#Missing	#Missing	#Missing	#Missing	#Missing	#Missing
Product	15,933	11,500	4,433	16,399	12,700	3,699

Central Sales

	Qtr1			Qtr2		
	Actual	Budget	Variance	Actual	Budget	Variance
100	9,246	6,500	2,746	9,974	7,300	2,674
200	7,269	6,800	469	7,440	7,000	440
300	10,405	6,200	4,205	10,784	6,800	3,984
400	10,664	5,200	5,464	11,201	5,800	5,401
Product	37,584	24,700	12,884	39,399	26,900	12,499

```
                                  Market Sales

                        Qtr1                                Qtr2
                  Actual   Budget  Variance   Actual  Budget  Variance
                ======== ======== ======== ======== ======== ========
100               32,057   23,000    9,057   34,279   25,600    8,679
200               27,443   20,000    7,443   28,196   20,900    7,296
300               30,126   21,500    8,626   31,893   23,400    8,493
400               23,792   13,200   10,592   25,045   15,300    9,745
  Product        113,418   77,700   35,718  119,413   85,200    34,21
```

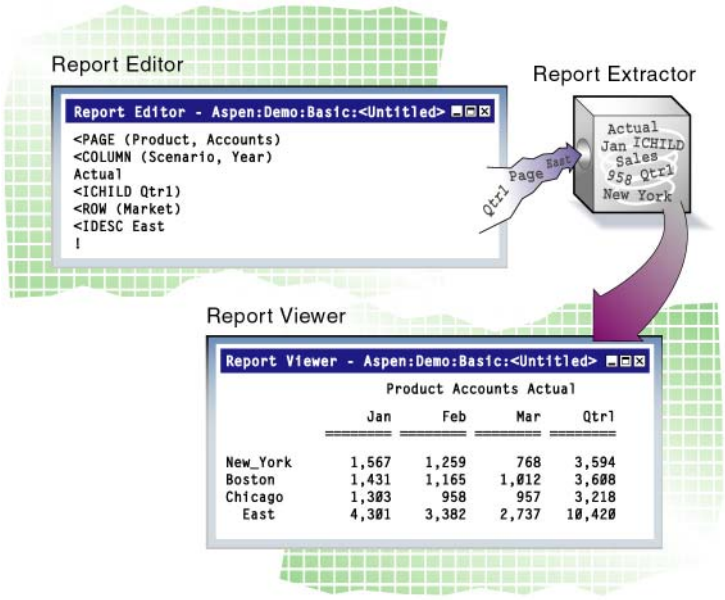
Understanding How Report Writer Works

The Report Writer consists of three main components:

- *Report Script Editor* is a text editor that you use to write the report script. In Report Script Editor, you use report commands to define formatted reports, export data subsets from a database, and produce free-form reports. You can then execute the script to generate a report. Report Script Editor features a text editing window, a customized right-click menu, a toolbar, and a message panel. Saved report scripts have the file extension `.rep`.
- *Report Extractor* retrieves the data information from the Analytic Services database when you run a report script.

- *Report Viewer* displays the complete report. Saved reports have the file extension `.rpt`.

Figure 205: Report Writer Components



Report Extractor

The Report Extractor processes the report script and retrieves data in the following order:

1. Composes the member list, based on all possible member combinations. For example, the following command retrieves member East and all of its descendants:

   ```
   <IDESCENDANTS East
   ```

2. Applies member restrictions. For example, the following command refines the member selection:

   ```
   <LINK
   ```

3. Orders the member output. For example, the following command determines the order in which members are sorted:

   ```
   <SORT
   ```

4. Extracts data from the following areas:

   - Local regions
   - Partitioned regions
   - Dynamically calculated data

5. Restricts data. For example, the following command suppresses the display of all rows that contain only missing values:

   ```
   {SUPMISSINGROWS}
   ```

6. Sorts data. For example, the following command returns rows with the highest values of a specified data column:

   ```
   <TOP
   ```

7. Formats output. For example, the following command skips one or more lines in the final output report:

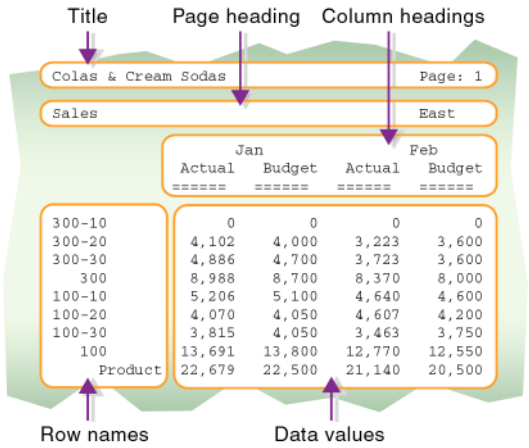
   ```
   {SKIP}
   ```

The order in which the Report Extractor retrieves data is important when using complex extraction and formatting commands. For example, because the Report Extractor restricts data (step 5) before sorting data (step 6), if you place conditional retrieval commands in the wrong order, the report output results can be unexpected. Be aware of the data retrieval process when designing report scripts.

Parts of a Report

Understanding the parts of a report is essential as you plan and design your own reports.

Figure 206: Elements of a Typical Report



A typical report is composed of the following parts:

- Page Headings list dimensions represented on the current page. All data values on the page have the dimensions in the page heading as a common property.

  ```
  <PAGE (Market, Measures)
  ```

- *Column Headings* list members across a page. You can define columns that report on data from more than one dimension, which results in *nested column headings*.

  ```
  <COLUMN (Year, Scenario)
  ```

- Row Headings list members down a page. You can define a member list that includes rows from more than one level within a dimension or from more than one dimension. The rows are indented below the dimension name.

  ```
  <ROW (Product)
  ```

- Titles contain user-defined text, date and time stamp, the user ID of the person running the report, page numbers, the name of the source database, or any other descriptive information. Titles are user-generated and optional, whereas page, column, and row headings are automatically generated, because they are necessary to clearly describe the data on the report page.

```
{ STARTHEADING
TEXT   1 "Prepared by:"
      14 "*USERNAME"
       C "The Electronics Club"
      65 "*PAGESTRING"
TEXT  65 "*DATE"
SKIP
ENDHEADING }
```

- Data values are the values contained in the database cells; they are the lookup results of member combinations or the results of calculations when the report is run through the Report Extractor. Each data value is the combination of the members in the page heading, column heading, and row name.

  All data values in a row share the properties of the row names of that row. A report can have zero or more row name dimensions, each of which produces column of row names, with the innermost row name column cycling the fastest.

Parts of a Report Script

A report script consists of a series of Report Writer commands, terminated by the bang (**!**) report output command.

You can enter one or more report scripts in a report script file. A report script file is a text file that you create with Report Script Editor or any text editor.

To build a report script, enter or select commands that define the layout, member selection, and format in Report Script Editor. The different elements of a script are color-coded to aid in readability. You can enable syntax auto-completion to help build scripts quickly.

The commands in Report Writer perform two functions, data extraction and formatting:

- Extraction commands deal with the selection, orientation, grouping, and ordering of raw data extracted from the database. These commands begin with less than signs (<).

- Formatting commands allow for customization of the report format and appearance, the creation of new columns, and calculation of columns and rows. These commands are generally contained within braces ({}), although some begin with less than signs (<).
- The *bang* character (**!**) terminates a series of commands and requests information from the database. You can place one or more report scripts, each terminated by its own ! command, in the same report file.

See the *Technical Reference* for detailed information about the various report commands that you can use.

Planning Reports

Report design is an important part of presenting information. Designing a report is easy if you include the proper elements and arrange information in an attractive, easy-to-read layout.

➤ To plan a report, perform the following tasks:

1. Consider the reporting needs and the time required to generate the report.
2. Make a rough sketch of the report. Be sure to include the following items:
   - Report layout
   - Number of columns
   - Members to include
   - Titles, if applicable
   - Format of the data values
3. Review the sketch; if you need to add additional data or formatting to the report, it is often apparent at this stage.
4. Determine ways to optimize the run time of the report.

   See Chapter 56, “Optimizing Reports and Other Types of Retrieval” for a comprehensive discussion of how to optimize a report script.

Note: As you plan the report, minimize use of numeric row names. To avoid ambiguity, give the rows names that describe their content.

Considering Security and Multiple-User Issues

You must use Essbase Administration Services in order to use Report Script Editor to create or modify a report script. You can also use any text editor to create script files. If you use Report Script Editor, it lets you create and modify report scripts stored on your desktop machine, as well as the Analytic Server. To modify report scripts stored on the server, you must have Application Designer or Database Designer access.

Analytic Services supports concurrent, multiple-user database access. As in most multiple-user environments, Analytic Services protects critical data with a security system. Users can read or update data only if they have the correct permissions.

When you execute a report script, the Analytic Services security system verifies that you have Read or higher access level to all data members specified in the report. In a filtering process identical to the one for retrieving members into a spreadsheet, Analytic Services filters any member from the output for which you have insufficient permissions.

To users who are only reporting data, locks placed by other users are transparent. Even if a user has locked and is updating part of the data required by the report, the lock does not interfere with the report in any way. The data in the report reflects the data in the database at the time you run the report. Running the same report later reflects any changes made after the last report ran.

See Chapter 36, “Managing Security for Users and Applications” for a comprehensive discussion of the Analytic Services security system.

Reviewing the Process for Creating Report Scripts

This section describes the process for creating a report script.

1. Create the report script. See “Creating Report Scripts” on page 667.
2. Check the report script syntax. See “Checking Script Syntax” in the *Essbase Administration Services Online Help*.
3. Save the report script. See “Saving Report Scripts” on page 667.
4. Run the report script. See “Executing Report Scripts” on page 668.
5. If desired, save the report. See “Saving Reports” in the *Essbase Administration Services Online Help*.

Creating Report Scripts

You can report on the data in a database using any of the following methods:

- Report Script Editor. Use Report Script Editor to create large-scale reports consisting of many pages of multidimensional data. Reports of this scale often can exceed the capabilities of even the most robust spreadsheet. Report Writer commands let you define formatted reports, export data subsets from an Analytic Services database, and produce free-form reports. To create a report script using Report Script Editor, see "Creating Scripts" in the *Essbase Administration Services Online Help*.
- A text editor.
- Through a spreadsheet. You can use report commands in a spreadsheet in free-form mode or in template retrieval mode. For information on reporting through the spreadsheet interface, see *Essbase Spreadsheet Add-in User's Guide*.
- Analytic Services APIs. See "Generating Reports Using the C, Visual Basic, and Grid APIs" on page 720 for more information about using report APIs or see the *API Reference* for syntax and technical descriptions.
- Third-party reporting tools.

For more information about creating and editing report scripts in Essbase Administration Services, see "About Report Script Editor" in *Essbase Administration Services Online Help*.

Saving Report Scripts

You can save a report script in the following locations:

- As a file on a client machine or network.
- As an object on Analytic Server. If you want other users to have access to the report script, save it on Analytic Server. You can associate the script object with the following objects:
  - An application and all the databases within the application, which lets you run the script against any database in the application.
  - A database, which lets you run the script against the specified database.

Report scripts have a `.rep` extension by default. If you run a report script from Essbase Administration Services, the script must have a `.rep` extension.

➤ To save a report script using Report Script Editor, see "Saving Report Scripts" in *Essbase Administration Services Online Help*.

Executing Report Scripts

When you execute a report script using Essbase Administration Services, you can send the results to the Report Viewer window, to a printer, and/or to a file. From the Report Viewer window, you can print, save, and copy the report.

Using Essbase Administration Services, you can execute a report in the background so that you can continue working as the report processes. You can then check the status of the background process to see when the report has completed.

For more information, see "Executing Report Scripts" in *Essbase Administration Services Online Help*.

Copying Report Scripts

You can copy report scripts to applications and databases on any Analytic Server, according to your permissions. You can also copy scripts across servers as part of application migration.

➤ To copy a report script, use any of the following methods:

Tool	Topic	Location
Administration Services	Copying Scripts	*Essbase Administration Services Online Help*
MaxL	**alter object**	*Technical Reference*
ESSCMD	COPYOBJECT	*Technical Reference*

Developing Free-Form Reports

Free-form reports are often easier to create than structured reports. The free-form reporting style is ideal for ad hoc reporting in the Report Script Editor window.

A free-form report does not include PAGE, COLUMN, or ROW commands and instead gathers this information from a series of internal rules that are applied to the report script by the Report Extractor when you run the report.

The following example script and report illustrate free-form reporting:

```
Sales Colas
Jan Feb Mar
Actual Budget
Illinois
Ohio
Wisconsin
Missouri
Iowa
Colorado
{UCHARACTERS}
Central
     !
```

This example produces the following report:

```
                                Sales 100

                 Jan                 Feb                 Mar
           Actual   Budget     Actual   Budget     Actual   Budget
         =======  =======    ======   ======     ======   ======
Illinois     829      700        898      700        932      700
Ohio         430      300        397      300        380      300
Wisconsin    490      300        518      400        535      400
Missouri     472      300        470      300        462      300
Iowa         161        0        162        0        162        0
Colorado     643      500        665      500        640      500
========     ===      ===        ===      ===        ===      ===
 Central   3,025    2,100      3,110    2,200      3,111    2,200
```

You can use formatting commands to add specific formats to a free-form report. The rest of the report is automatically produced in a format similar to that of any other report. When PAGE, COLUMN, and ROW commands are omitted, Analytic Services formats free-form reports according to the following rules:

1. The Report Extractor finds the last member or members of a single dimension defined in the report specification (before the report output operator !). This dimension becomes the ROW dimension for the report. All remaining selections become PAGE or COLUMN dimensions, as defined by rules 2 and 3.
2. The Report Extractor searches for any single-member selections. If a single member is found that does not satisfy rule 1, that dimension becomes a PAGE dimension.
3. The Report Extractor searches for all remaining dimension members that do not satisfy rules 1 or 2. If any remaining members are found, those dimensions become COLUMN dimensions. COLUMN dimensions are nested in the order of selection in the free-form script.
4. The Report Extractor searches the database outline for any dimensions not specified in the report specification. If any unspecified dimensions are found, they become PAGE dimensions (the default for single-member selections, as defined in rule 2).
5. A subsequent selection of one or more consecutive members from a given dimension overrides any previous selection for that dimension.

For example, the following report recognizes California, Oregon, Washington, Utah, Nevada, and West as members of Market.

```
Sales
Jan Feb Mar
Actual Budget
Apr May Jun
California
Oregon
Washington
Utah
Nevada
{UCHARACTERS}
West
    !
```

The Report Extractor applies free-form formatting rules to this report as follows:

1. Because California, Oregon, Washington, Utah, Nevada, and West are listed last, the Report Extractor treats them as if ROW(Market) had been specified (according to rule 1).
2. Sales is a single-member selection from dimension Measures. The Report Extractor treats this member as if PAGE(Measures) had been specified (according to rule 2).
3. After searching the remaining members, the Report Extractor finds members of dimensions Year and Scenario, which it treats as COLUMN(Year, Scenario), according to rule 3.
4. The Report Extractor searches the database outline and finds that dimension Product is not specified in the report specification. Since Product is a single-member selection, the Report Extractor treats this member as if PAGE (Product) had been specified (according to rule 4).
5. Finally, the Report Extractor finds that Apr May Jun is from the same dimension as Jan Feb Mar and is displayed on a subsequent line of the script. The Report Extractor discards the first specification (Jan Feb Mar) and uses the second (Apr May Jun).

As a result, the report example produces the following report:

```
                                Product Sales

                      Actual                          Budget
                 Apr      May      Jun      Apr      May      Jun
              =======  ======   ======   ======   ======   ======
California    3,814    4,031    4,319    3,000    3,400    3,700
Oregon        1,736    1,688    1,675    1,100    1,000    1,100
Washington    1,868    1,908    1,924    1,500    1,600    1,700
Utah          1,449    1,416    1,445      900      800      800
Nevada        2,442    2,541    2,681    1,900    2,000    2,100
======        =====    =====    =====    =====    =====    =====
  West       11,309   11,584   12,044    8,400    8,800    9,400
```

Note: You cannot use substitution variables in free-form mode.

Chapter

32

Developing Report Scripts

Once you understand the basics of creating report scripts, you can create more complex reports. You create a report using *extraction commands* which specify member combinations for pages, columns, and rows. You use *formatting commands* to determine the visual design of the report and to control the display of the data values. Formatted data values are displayed in the report when you run the script, based on the combined extraction and report commands.

This chapter provides information about creating complex report scripts:

For fundamental information about reports and report scripts, see Chapter 31, "Understanding Report Script Basics."

For report command syntax and an extensive set of report script examples, see the *Technical Reference*.

Understanding Extraction and Formatting Commands

Extraction commands perform the following actions:

- Determine the selection, orientation, grouping, and ordering of raw data records extracted from the database. Extraction commands are based on either dimension or member names, or keywords. Their names begin with the greater-than symbol (>).
- Apply to the report from the line on which they occur until the end of the report. If another extraction command occurs on a subsequent line of the report, it overrides the previous command.

Formatting commands perform the following actions:

- Let you customize the format and appearance of a report and create report-time calculations. Formatting commands are generally enclosed inside left and right braces ({ }), although there are several formatting commands that begin with the less-than (<) character.
- Are either applied globally within the report script or are specific to a member.

Understanding Report Script Syntax

To build a report, you enter commands that define the layout, member selection, and format you want in Report Script Editor. The different elements of a script are color-coded to aid in readability. You can enable auto-completion to help build scripts interactively as you type. For more information, see “About Report Script Editor” in *Essbase Administration Services Online Help*.

When you write a report script, follow these guidelines:

- Separate commands with at least one space, tab, or new line for readability. Report processing is not affected by extra blank lines, spaces, or tabs.
- Enter commands in either uppercase or lowercase. Commands are not case-sensitive. If the database outline is case-sensitive, the members in the report script must match the outline.
- To start report processing, enter the ! (bang) report output command 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 braces. For example, these formats are synonymous:

```
{UDATA SKIP}
{UDATA} {SKIP}
```

- Enclose member names in quotation marks in the following cases:
  - Names beginning with an ampersand (for example, “&Product”).
  - Names containing spaces (for example, “Cost of Goods Sold”).
  - Names containing the word Default (for example, “Default Value”).
  - Names containing one or more numerals at the beginning of the name (for example, “100-Blue”)
  - Names containing any of the following characters:

*	asterisks	-	dashes, hyphens, or minus signs
@	at signs	<	less than signs
{}	braces	()	parentheses
[]	brackets	+	plus signs
,	commas	;	semicolons
:	colons	/	slashes

- If a formatting command is preceded by three or more underscore, equal sign, or hyphen characters, respectively, 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, so you can include multi-line comments.
- Exercise caution if you abbreviate command names. Remember that many names begin with the same letters, and the results may be unexpected unless you use a completely unambiguous abbreviation.

Designing the Page Layout

Reports are two-dimensional views of multidimensional data. You can use page layout commands to incorporate additional dimensions that are defined as nested groups of columns or rows on a page, or additional pages in the report.

The page layout is composed of headings that make up the columns and rows of a page. You define the basic layout of a report using page, row, and column data extraction commands combined with specific member selections.

Each component of page layout has a different formatting command:

<PAGE <COLUMN <ROW

In addition, the <ASYM and <SYM commands override the default method of interpreting the column dimension member lists, and produce either an asymmetric or symmetric report format.

For information about formatting the page, column, or row headings, see "Formatting Page, Column, and Row Headings" on page 682.

Creating Page, Column, and Row Headings

➤ To design the report headings, perform the following tasks:

1. Type `<PAGE(dimensionname, dimensionname)`

   where *dimensionname* lists dimensions represented on the current page. All data values on the page have the dimensions in the page heading as a common property. The following is a valid <PAGE command:

   ```
   <PAGE (Measures, Market)
   ```

2. Press **Enter**.
3. Type `<COLUMN(dimensionname)`

   where *dimensionname* equals the name of each dimension to display across the page. The following is a valid <COLUMN command:

   ```
   <COLUMN (Year)
   ```

   You can add dimension names to create nested column headings.

4. Press **Enter**.

5. Type <ROW(*dimensionname*)

   where *dimensionname* equals the name of each dimension to display down the page. The following is a valid <ROW command:

   ```
   <ROW (Market)
   ```

6. Press **Enter**.

Note: You can select additional members to associate with the heading commands. If you type a member name as a parameter for the PAGE, COLUMN, or ROW commands, Report Extractor automatically associates the member with the appropriate dimension.

The following report script is based on the Sample Basic database:

```
<PAGE (Product, Measures)
<COLUMN (Scenario, Year)
Actual
<ICHILDREN Qtr1
<ROW (Market)
<IDESCENDANTS East
    !
```

This script produces the following report:

```
              Product Measures Actual

                    Jan      Feb      Mar     Qtr1
               ======== ======== ======== ========
New York             512      601      543    1,656
Massachusetts        519      498      515    1,532
Florida              336      361      373    1,070
Connecticut          321      309      290      920
New Hampshire         44       74       84      202
  East             1,732    1,843    1,805    5,380
```

You can create page, column, and row headings with members of attribute dimensions. The following report script is based on the Sample Basic database:

```
<PAGE (Measures,Caffeinated)
Profit
<COLUMN (Year,Ounces)
Apr May
"12"
```

```
<ROW (Market,"Pkg Type")
Can
<ICHILDREN East
    !
```

This script produces the following report:

```
                          Profit Caffeinated 12 Scenario
                                      Apr      May
                                 ======== ========
New York          Can                276      295
Massachusetts     Can                397      434
Florida           Can                202      213
Connecticut       Can                107       98
New Hampshire     Can                 27       31
  East            Can              1,009    1,071
```

Modifying Headings

You can perform the following modifications to headings in the report:

Task	Report Command
Create a custom 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. Use the ENDHEADING command to specify the end of the custom heading.	STARTHEADING
Display the page heading, either the default heading or the heading as defined with the STARTHEADING and ENDHEADING commands. Use this command to re-enable the page heading display if the SUPHEADING command has been used.	HEADING
Force the immediate display of the heading without waiting for the next unsuppressed data row, when heading suppression commands are in use.	IMMHEADING
Automatically turn on the display of the column header.	COLHEADING
Display the page heading before the next data output row.	PAGEHEADING

For descriptions of the SUPPRESS commands used to suppress headings, see "Suppressing Page, Column, and Row Formatting" on page 685.

Creating Symmetric and Asymmetric Reports

Analytic Services reports can contain symmetric or asymmetric column groups. Analytic Services determines the symmetry of column groups automatically, based on the members you select.

A *symmetric report*, shown below, is characterized by repeating, identical groups of members.

```
                East                              West
    Budget          Actual            Budget          Actual
Q1   Q2   Q3     Q1   Q2   Q3       Q1   Q2   Q3    Q1   Q2   Q3
```

An *asymmetric report*, shown below, is characterized by groups of nested members that differ by at least one member in the nested group. There can be a difference in the number of members or the names of members.

```
                East                              West
    Budget          Actual                        Budget
Q1   Q2   Q3     Q1   Q2   Q3                  Q1   Q2   Q3
```

By default, Analytic Services creates a symmetric report unless you select the same number of members for all column dimensions.

For an example of an asymmetric report, see "Sample 13, Creating Asymmetric Columns," in the Examples of Report Scripts page of the *Technical Reference*.

The Analytic Services evaluation of symmetry versus asymmetry takes place prior to any ordering, restriction on columns, or application of the effects of calculated columns.

Overriding Default Column Groupings

You can override the default column grouping that Analytic Services selects for reports with the <SYM and <ASYM commands. <SYM and <ASYM affect the member selection commands that follow them in a report.

1. Use the <SYM command when the selection of column members meets the requirements of the rule for asymmetry, but you want to produce a symmetric report. The <SYM command always produces a symmetric report, creating all combinations of each column dimension.
2. Turn off the symmetric format and restore the rules for asymmetric reports with the <ASYM command.

Changing Column Headings

If you only need to change the column headings rather than the symmetry of the report, the <PYRAMIDHEADERS and <BLOCKHEADERS formatting commands are useful.

- Use the <BLOCKHEADERS formatting command to change the pyramid-style headers used in symmetric reports to block-style headers like those used in asymmetric reports. A symmetric report uses the <PYRAMIDHEADERS mode of column layout by default.
- Use the <PYRAMIDHEADERS formatting command to change the block-style headers used in asymmetric reports to pyramid-style headers like those used in symmetric reports. An asymmetric report uses the <BLOCKHEADERS mode of column layout.

Formatting

Formatting commands define the format of data and labels in the final report. These commands are generally enclosed in left and right braces ({}).

The two types of formatting commands are *global* and *member-specific* commands.

- Global commands are executed when they occur in the report script file, and stay in effect until the end of the report file or until another global command replaces them.

  For example, the {SUPMISSINGROWS} command suppresses all rows in the report script file that contain only missing values.

- Member-specific commands are executed as they are encountered in the report script, usually the next member in the report script, and affect only that member. A format attached to a member is executed before that member is processed.

  For example, the {SKIP} command skips the specified number of rows between row dimensions in a report script. If you want additional rows to skip lines, you must use the SKIP command again.

Formatting Report Pages

There are a number of formatting commands that you can use to design the look of the final report pages.

See the *Technical Reference* for report formatting examples.

Setting Page Length and Width and Centering

You can set the following page specifications in the report script:

Task	Report Command
Specify the column widths in a report.	WIDTH
Set the left margin of the report.	LMARGIN
Set the center of the page.	SETCENTER

Inserting Page Breaks

You can set the following types of page breaks in the report script:

Task	Report Command
Set the number of lines for each page.	PAGELENGTH
Force a page break regardless of how many lines have been generated for the current page.	NEWPAGE

Task	Report Command
Insert a page break whenever a member from the same dimension as the member in the command changes from one line in the report to the next. Use the NOPAGEONDIMENSION command to turn off this function.	PAGEONDIMENSION
Enable page breaks in a report when the number of lines on a page is greater than the current PAGELENGTH setting.	FEEDON

Formatting Page, Column, and Row Headings

Column and row formatting commands make up a special type of format setting commands.

Specifying Column Formats

Specifications for column formatting commands can precede or follow the columns to which they apply, depending on the desired format.

For example, in the following script, based on the Sample Basic database, the first {DECIMAL} command is processed after only two columns are set up, Actual and Budget. The {DECIMAL} command, however, refers to a column three, which does not yet exist. Analytic Services responds to this command by dynamically expanding the report to three columns. When the report specification expands to six columns, the {DECIMAL} formatting applies to columns three *and* six (and all multiples of three).

Analytic Services performs this pattern extension on the assumption that when another dimension is added, causing repetitions of previous column dimension groups, the formatting should repeat as well. The second {DECIMAL} formatting command is then applied to columns 1 and 4 only, as it occurs after the creation of six columns.

```
<PAGE (Measures, Market)
Texas Sales
     <COLUMN (Scenario, Year)
     Actual Budget
{DECIMAL 2 3 }
     Jan Feb Mar
{DECIMAL 1 1 4 }
```

```
<ROW (Product)
<DESCENDANTS "100"
    !
```

This script produces the following report:

```
                          Sales Texas

                 Actual                        Budget
          Jan      Feb      Mar      Jan      Feb      Mar
          ===      ===      ===      ===      ===      ===
100-10   452.0     465    467.00     560.0     580    580.00
100-20   190.0     190    193.00     230.0     230    240.00
100-30 #MISSING #MISSING #MISSING #MISSING #MISSING #MISSING
```

The following scripts demonstrate two approaches to column formatting that produce identical results. In the first script, the first two {DECIMAL} commands are positioned to format every first and third column by distributing the formats when Jan Feb displays *after* processing the {DECIMAL} command. These examples are based on the Sample Basic database.

```
//Script One: Format Columns by Distributing the Formats

<PAGE (Measures, Market)
California Sales
     <COLUMN (Scenario, Year)
     Actual Budget Variance
{DECIMAL 1 1 }
{DECIMAL 2 3 }
     Jan Feb
     //     {DECIMAL 1 1 4 }   These lines are commented; the
     //     {DECIMAL 2 3 6 }   Report Extractor ignores them.
<ROW (Product)
<DESCENDANTS "100"
    !
```

The two {DECIMAL} commands are positioned to format the individual columns 1, 3, 4, and 6.

```
//  Script Two: Format Columns by Direct Assignment

<PAGE (Measures, Market)
California Sales
     <COLUMN (Scenario, Year)
```

```
     Actual Budget Variance
     //     {DECIMAL 1 1 }     These lines are commented; the
     //     {DECIMAL 2 3 }     Report Extractor ignores them.
     Jan Feb
{DECIMAL 1 1 4 7 }
{DECIMAL 2 3 6 9 }
<ROW (Product)
<DESCENDANTS "100"
    !
```

Both scripts produce the following report:

```
                          Sales California

              Actual           Budget            Variance
           Jan      Feb     Jan       Feb     Jan       Feb
         =====    ====    ====      ====    =====      ====
 100-10   678.0     645   840.00    800.0    (162)  (155.00)
 100-20   118.0     122   140.00    150.0     (22)   (28.00)
 100-30   145.0     132   180.00    160.0     (35)   (28.00
```

Note: 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.

Accommodating Long Column Names and Row Names

Member names that are too long to fit into the column are automatically truncated; the tilde character (~) signifies that part of the name is missing. Long member names are common when using aliases in the report.

There are several ways to modify columns to display the entire member name.

Task	Report Command
Define the column width for all row members in the column.	NAMEWIDTH
Change where the row member column is displayed, and to shift the remaining columns left or right to make room fro the long member names.	NAMESCOL

Suppressing Page, Column, and Row Formatting

You can suppress the display of page heading, columns, and rows in a report by using SUPPRESS commands.

Suppress	Report Command
The default column heading in the report.	SUPCOLHEADING
Rows that have only zero or missing values.	SUPEMPTYROWS
All rows that contain missing values. Use INCMISSINGROWS, INCZEROROWS, or INCEMPTYROWS to display rows that are empty, or have missing data or zeros.	SUPMISSINGROWS
The page member heading whenever a heading is generated.	SUPPAGEHEADING
The page and column headings, all member names, page breaks, commas, and brackets in the final report. To turn on the display of columns of row member names, use the NAMESON command. To turn on the use of commas and brackets, use the COMMAS and BRACKETS commands.	SUPALL
The default heading (page header and column headers) or custom header, if defined, at the top of each page.	SUPHEADING
Row member names in the final report. Use the NAMESON command to include row member names in the report.	SUPNAMES
All output while continuing to process all operations, such as calculations, format settings, and so forth. Use the OUTPUT command to reverse the actions of SUPOUTPUT.	SUPOUTPUT
Currency information when you use the CURRENCY command to convert the data values in a report to a specified currency.	SUPCURHEADING

Repeating Row Names

To repeat the row member names on every line of the report, use the <ROWREPEAT command. Use the <NOROWREPEAT command to prevent 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 enabled by default.

Using Tab Delimiters

You can place tabs between columns rather than spaces in report scripts, for example, when you want to export report output into another form.

To replace spaces with tab delimiters, type {TABDELIMIT} anywhere in the report script.

Adding Totals and Subtotals

Column and row calculations let you create additional calculations that are not defined as part of the database outline. For example, you can use column and row calculations to create extra columns or rows in a report, based upon selected data members, and perform calculations on these or existing columns and rows.

For examples of report scripts that contain column and row calculations, see "Sample 14, Calculating Columns" in the "Example of Report Scripts" page of the *Technical Reference*.

Totaling Columns

The CALCULATE COLUMN command lets you create a new report column, perform on-the-fly calculations, and display the calculation results in the newly created column.

The following table summarizes column calculation commands:

Task	Report Command
Create a new report column, perform dynamic calculations, and display the calculation results in the newly created column. Use the OFFCOLCALCS command to temporarily disable column calculations in the report, and ONCOLCALCS to re-enable calculations.	CALCULATE COLUMN
Remove all column calculation definitions from the report.	REMOVECOLCALCS

CALCULATE COLUMN adds up to 499 ad hoc column calculations to a report. Each new calculated column is appended to the right of the existing columns in the order in which it is created, and given the next available column number. These columns calculate the sum of data across a range of columns or an arithmetic expression composed of simple mathematical operators.

The CALCULATE COLUMN command supports the standard mathematical operations. For syntax and parameter descriptions, see the *Technical Reference*.

If you use the same name for more than one column, Analytic Services creates only the last column specified in the CALCULATE COLUMN command. Use a leading space with the second name (and two leading spaces with the third name, and so on) to create a unique column name.

Alternately, you can add descriptive text far enough to the right that it is truncated to the column width. You can, for example, use the names Q1 Actual and Q1 Budget to distinguish similar column names without affecting the appearance of the report. Column names are printed with right justification until the column header space is filled. Excess characters are then truncated to the right.

Divide lengthy column name labels into two or more lines. The maximum number of lines across which you can divide a label is equal to the number of column dimensions designated in the report specification. To break a column name, insert the tilde character (~) in the name at the point where you want the break. You must also specify at least two members for each column dimension to use the maximum number of lines.

This example is based on the Sample Basic database.

```
{CALCULATE COLUMN "Year to Date~Actual Total" = 1 : 2}
{CALCULATE COLUMN "Year to Date~Budget Total" = 3 : 4}
```

The example produces the following report:

```
                                    Sales East
                Actual   Year to Date      Budget    Year to Date
              Jan    Feb  Actual Total    Jan    Feb  Budget Total
            ===== ====== ============= ====== ====== =============
400-10        562    560         1,122    580    580         1,702
400-20        219    243           462    230    260           722
400-30        432    469           901    440    490         1,391
```

As a rule, in symmetric reports, if a calculated column name has fewer levels than the number of column dimensions, the previous member (to the left) of each of the column dimensions, above the top level supplied in the calculated column name, is attributed to the calculated column. If normal PYRAMIDHEADERS mode is in use, the centering of those higher-level column members shifts to the right to include the calculated column or columns. Column header members on the same level as calculated column names are not applied to the new calculated column or columns, and their centering does not shift.

If BLOCKHEADERS mode is in use, that is, if every member applying to a column is repeated above that column, the same rules apply, except that instead of shifting column header member centering, they are repeated in the appropriate higher levels of the calculated column name.

Asymmetric reports do not have groups of columns that share a member property. These reports still allow multiple-level column names up to the number of column levels defined, but member properties from preceding columns are not automatically shared and used for those levels that are not defined.

In cases where there are fewer column header dimensions than the number of levels that you want, you can create multi-line column labels. In this case, use TEXT, STARTHEADING, ENDHEADING, and other formatting commands to create a custom heading.

For the syntax and definitions of column calculation commands, see the *Technical Reference*.

Numbering Columns

If the number of regular (not calculated) columns varies in the report because multiple sections in the report have different numbers of columns, the column numbers used to identify the calculated columns shift accordingly, as illustrated in the following examples:

- If the first section of a report has 12 columns (including row name columns), and 3 calculated columns are declared, column numbers 0–11 are the regular columns, and columns 12–14 are the calculated columns.
- If a second section of the report reduces the number of regular columns to 6, then the regular columns are columns 0–5, and the same calculated columns are columns 6–8.
- Likewise, if the number of regular columns is increased, the numbering of the calculated columns starts at a higher number.

In the example, CC1, CC2, and CC3 represent the names of three calculated column names. The column numbering for a report with two different sections with varying numbers of regular columns looks as follows:

```
internal
col # s: 0     1      2      3      4      5      6      7
             Jan    Feb    Mar    Apr    CC1    CC2    CC3
             ===    ===    ===    ===    ===    ===    ===
   Sales       1      3      5      3     22     55     26
   Expense     1      2      5      3     23     65     33

                  same report- new section
internal
col # s: 0     1      2      3      4      5
           Qtr1    YTD    CC1    CC2    CC3
            ===    ===    ===    ===    ===
   Sales       2      9     22     57     36
   Expense     4      8     56     45     33
```

If you do not want the calculated columns in the second section, or if you need a different set of column calculation, use the command REMOVECOLCALCS to clear the old ones out. You can then define new column calculations.

This example assumes that all three column calculations had no references to regular columns other than columns 1 and 2. If CC3's calculation were = 1 + 3 + 6, when the second section of the report starts, an error occurs stating that the column calculation referred to a nonexistent column (6).

Totaling Rows

Row calculations create summary rows in a report. You can use *summary rows* to calculate the sum of data across a range of rows or to calculate an arithmetic expression composed of simple mathematical operators.

The following table summarizes row calculation commands:

Task	Report Command
Create a new row and associate it with a row name or label. This process is similar to declaring a variable. You can also perform simple calculations with CALCULATE ROW. For more complex calculations, use SETROWOP. See also OFFROWCALCS and ONROWCALCS.	CALCULATE ROW
Temporarily disable row calculations in the report. See also CALCULATE ROW and ONROWCALCS.	OFFROWCALCS
Re-enable calculations after using OFFROWCALCS. See also CALCULATE ROW and OFFROWCALCS.	ONROWCALCS
Define complex calculations for the row specified in CALCULATE ROW. SETROWOP defines a calculation operator to be applied to all subsequent output data rows. You can display the calculation results in the newly created row with the PRINTROW command.	SETROWOP
Immediately display the row specified in CALCULATE ROW to the report.	PRINTROW
Reset the value of the calculated row to #MISSING. See also CLEARALLROWCALC.	CLEARROWCALC
Reset the value of all calculated rows after using the CLEARROWCALC command.	CLEARALLROWCALC

Task	Report Command
Create a new calculated row with captured data. See also SAVEANDOUTPUT.	SAVEROW
Capture data and output the result after using the SAVEROW command.	SAVEANDOUTPUT

For the syntax and definitions of row calculation commands, see the *Technical Reference*.

Commands that designate columns must use valid data column numbers, as determined by the *original* order of the columns.

- Precede and follow all operators in an expression with a single space.
- Analytic Services does not support nested (parenthetical) expressions.
- Analytic Services supports integer and floating-point constants in expressions as single entries or members of an array.

The CALCULATE ROW command can specify an operation (+, -, *, /, or OFF) as an equation consisting of constants, other calculated rows, and operators. Equations are evaluated at the time of declaration. Member names are not supported in expressions with the CALCULATE ROW command.

If you specify an operator, the operator applies to subsequent output rows and stores the result in the calculated row. Specifying an operator is useful for aggregating a series of rows to obtain a subtotal or total. To reset the operator, use SETROWOP. If the CALCULATE ROW command does not specify either an equation or an operator, the + operator is assumed.

The CALCULATE ROW command supports the standard mathematical operations. For syntax and parameter descriptions, see the *Technical Reference*.

This example is based on the Sample Basic database.

```
{ CALC ROW "Total Sales" = "Sales..Group1"
     + "Sales..Group2" }
```

The example creates “Total Sales” based on two other calculated rows.

Changing How Data Is Displayed

You can use a variety of formatting commands to customize how data displays in the final report.

Underlining

Use underlining as a visual aid to break up blocks of information in a report.

Task	Report Command
Set the default underline character to display in the report.	UNDERLINECHAR
Underline all characters that are not blank in the preceding row.	UCHARACTERS
Underline all the columns in the preceding row.	UCOLUMNS
Underline all the data columns for a row, while not underlining the row name columns.	UDATA
Underline all the row name columns in the preceding row while not underlining the data columns.	UNAME
Underline the row member names in a row whenever a member from the same dimension as the member in the command changes. Use the NOUNAMEONDIM command to turn off underlining for new rows.	UNAMEONDIMENSION

Suppressing Data Formatting

You can suppress data that you do not want to be displayed in the final report by using SUPPRESS commands.

Suppress	Report Command
Brackets around negative numbers. Use the BRACKETS command to re-enable brackets.	SUPBRACKETS
Commas in numbers greater than 999. Use the COMMAS command to re-enable commas.	SUPCOMMAS
Rows that have only zero data values. Use INCZEROROWS or INCEMPTYROWS to re-enable the display.	SUPZEROROWS
The European method for displaying numbers (2.000,01 whereas the non-European equivalent is 2,000.01). Use the EUROPEAN command to re-enable European number display.	SUPEUROPEAN
The automatic insertion of a page break whenever the number of lines on a page exceeds the current PAGELENGTH setting.	SUPFEED
Formats that produce output such as underlines and skips. Use INCFORMATS to re-enable the display.	SUPFORMATS
Text masks that were defined in the report using the MASK command. Use INCMASK to re-enable the display.	SUPMASK

See “Suppressing Page, Column, and Row Formatting” on page 685 for descriptions of the SUPPRESS commands used to suppress formats.

Indenting

Use indenting to provide visual clues to row levels of the script.

Task	Report Command
Shift the first row names column in column output order by a specified number of characters.	INDENT
Indent subsequent row members in the row names column based on the generation in the database outline. Use the NOINDENTGEN command to left-justify row member names based on generation name.	INDENTGEN

Inserting Custom Titles

Titles are user-generated and optional, in contrast to the automatically generated page and column headings and row names, which describe the data on the report page.

Titles repeat at the top of each report page, and provide the following information about a report:

- A date and time stamp
- The user ID of the person running the report
- Page numbers
- The name of the source database
- Any other descriptive information

To add a title to the report, use the TEXT command, combined with any of the following:

- Any of the predefined keywords that automatically display information in the report
- A text string that you define

Note: You can also use the TEXT command at the bottom of the report to provide summary information.

See the *Technical Reference* for the syntax and definitions of Report Writer commands.

Replacing Missing Text or Zeros with Labels

When you run a report, there are often many empty data cells where no data was applicable to the retrieval, or cells where the value is zero.

The report displays the default #MISSING label in the data cell when no data values are found.

➤ To replace the #MISSING label with a text label add the following to the report script:

At the point in the script where you want to replace the #MISSING label with a text label, type:

```
{MISSINGTEXT ["text"]}
```

where *text* is any text string that you want to display in the data cells.

You can place the MISSINGTEXT command at any point in the report script; the command applies throughout the script.

Note: You can also suppress #MISSING labels from appearing in the report. See "Suppressing Data Formatting" on page 693 for descriptions of SUPPRESS commands used to suppress labels, or see the *Technical Reference* for the syntax and definitions of Report Writer commands.

➤ To replace zeros with a text label add the following to the report script:

At the point in the script where you want to replace zeros with a text label, type

```
{ZEROTEXT ["text"]}
```

where *text* is any text string that you want to display in the data cells.

Note: If a value is equal to #MISSING, any string you try to insert after that value using the AFTER command does not print. AFTER strings also do not print if you replace #MISSING with some other value (such as 0).

Adding Blank Spaces

Adding blank spaces in a report draws the reader to key information, such as totals.

Task	Report Command
Add one or more blank lines in the final report.	SKIP
Add a blank line when a member from the same dimension as the specified member in the command changes on the next line in the report. Use the NOSKIPONDIMENSION command to turn off insertion of a new line.	SKIPONDIMENSION

Changing How Data Values Display

You can use the following commands to change how data values display in the final report.

Task	Report Command
Turn on the display of commas for numbers greater than 999 after commas have been suppressed with either a SUPCOMMAS or SUPALL command.	COMMAS
Turn on the display of brackets around negative numbers instead of negative signs, after using the SUPBRACKETS command earlier in the script.	BRACKETS
Include a percentage sign or other character after the data values.	AFTER
Include a dollar sign or other character before the data values.	BEFORE
Use the European method for displaying numbers where decimal points are used as the thousands separator character while commas separate the decimal portion of the number from the integer portion (2.000,01, whereas the non-European equivalent is 2,000.01).	EUROPEAN
Overwrite text in each output row with a specified characters and position.	MASK

Selecting and Sorting Members

The data that is displayed in the final report is based upon the members that you select and the order in which you display them. In addition, you can use conditional retrievals to further refine the selection and sorting of members.

Selecting Members

Member selection commands are extraction commands that select ranges of members based on database outline relationships, such as sibling, generation, and level. Using member selection commands ensures that any changes to the outline are automatically reflected in the report, unless you change the member name on which the member selection command is based. Attribute dimensions can be included in member selection commands.

Task	Report Command
Select all the members from the same dimension as the dimension member.	ALLINSAMEDIM
Include all the siblings of the specified member.	ALLSIBLINGS
Include all the ancestors of the specified member.	ANCESTORS
Select a base dimension member based on its attributes.	ATTRIBUTE
Select all members in the level immediately below the specified member.	CHILDREN
Include the descendants of the specified member to the report, excluding the dimension top.	DESCENDANTS
Select the level 0 members, that is, the members at the bottom of the dimension.	DIMBOTTOM
Include the top member of the dimension.	DIMTOP
Include a member and its ancestors.	IANCESTORS
Select the specified member and all members in the level immediately below it.	ICHILDREN
Include the specified member and its descendants.	IDESCENDANTS
Include the specified member and its parent.	IPARENT

Task	Report Command
Include all members from the same dimension and generation as the specified member.	OFSAMEGEN
Include all members from the same dimension and on the same level as the specified member.	ONSAMELEVELAS
Include the parent of the specified member to the report.	PARENT
Extract data for a specified date or for a time period before or after a specific date.	TODATE
Include base dimension members associated with the specified attribute dimension.	WITHATTR

Selecting Members by Using Generation and Level Names

Generation and level name selection commands identify a specific level or generation of members based on either of the following items:

- The default generation or level name in the outline
- The user-defined generation or level name in the outline

When you use generation and level names, changes to the outline are automatically reflected in the report. You can either define your own generation and level names, or you can use the default names provided by Analytic Services. For a discussion, including examples, of generations and levels, see "Generations and Levels" on page 36.

Using generation or level names whenever possible makes the report easier to maintain. Because you do not have to specify a member name in the report, you do not need to change the report if the member name is changed or deleted from the database outline.

Note: Generation and level names are stand-alone commands. You cannot use them in place of member names in report extraction or formatting commands. For example, you *cannot* use them with the <DESCENDANTS or <CHILDREN commands.

➤ To use default level names add the following to the report script:

At the point in the script where you want to select a member by the default level name, use the following format:

`Levn,dimensionName`

where *n* is the level number.

dimensionName is the name of the dimension from which you want to select the members.

Note: Do *not* put a space after the comma.

For example, Lev1,Year selects all the level 1 members of the Year dimension.

➤ To use default generation names add the following to the report script:

At the point in the script where you want to select a member by the default generation name, use the following format:

`Genn,dimensionName`

where *n* is the generation number.

dimensionName is the name of the dimension from which you want to select the members.

Note: Do *not* put a space after the comma.

For example, Gen2,Year selects all the generation 2 members of the Year dimension.

Note: These default generation and level names are not displayed in Outline Editor.

The following example is based on the Sample Basic database. It uses the default generation name Gen2,Year to generate a report that includes the members Qtr1, Qtr2, Qtr3, and Qtr4 from the Year dimension.

```
<PAGE(Product)
<COLUMN(Year)
<ROW (Measures)
{OUTALTNAMES}
Cola
Gen2,Year
Sales Profit
    !
```

The report script produces the following report:

```
                        Cola Market Scenario
                  Qtr1     Qtr2     Qtr3     Qtr4
               ======== ======== ======== ========
Sales            14,585   16,048   17,298   14,893
    Profit        5,096    5,892    6,583    5,206
```

Selecting Dynamic Time Series Members

You create and identify dynamic members in the database outline; they are members that are calculated only during user retrieval requests, such as generating a report script. The time dimension has special Dynamic Time Series option. The Dynamic Time Series option has reserved the following generation names that you can define in the outline alias table:

Generation Name	Reserved Names	Explanation
History	H-T-D	History-to-Date
Year	Y-T-D	Year-to-Date
Season	S-T-D	Season-to-Date
Period	P-T-D	Period-to-Date
Quarter	Q-T-D	Quarter-to-Date
Month	M-T-D	Month-to-Date
Week	W-T-D	Week-to-Date
Day	D-T-D	Day-to-Date

See "Applying Predefined Generation Names to Dynamic Time Series Members" on page 576 for information about creating and maintaining Dynamic Time Series generation names in the database outline.

Note: The database header message for the outline identifies the number of dynamic members that are enabled in the current outline.

➤ To select a Dynamic Time Series member add the following to the report script:

At the point in the script where you want to select a Dynamic Time Series member, use either of the following formats:

```
<LATEST memberName
```

where *memberName* is the name of the member in the time dimension.

The <LATEST command is a global command that is applied to the entire report script, and is an aggregation based on the lowest level member within the dimension.

```
reservedName(memberName)
```

where *reservedName* is the reserved Dynamic Time Series generation name, and the *memberName* is the name of the member in the time dimension.

If you use this syntax to specify a Dynamic Time Series, the time series name is associated only to the member listed in the argument.

When you run the report script, the members are dynamically updated, and the information is incorporated into the final report.

Note: You must type the Dynamic Time Series string exactly as it is displayed in the database outline; you cannot create your own string and incorporate it into the final report.

You can create an alias table for the Dynamic Time Series members in the database outline, and use the aliases instead of the predefined generation names.

Selecting Members by Using Boolean Operators

Boolean operators let you specify precise member combinations within a report, which is particularly useful when dealing with large outlines. Use the AND, OR, and NOT Boolean operators, combined with extraction commands, to refine member selections within the report script.

- Use the AND operator when all conditions must be met.
- Use the OR operator when one condition of several must be met.
- Use the NOT operator to choose the inverse of the selected condition.

➤ To create a Boolean expression using operators add the following to the report script:

At the point in the script where you want to use linking, enter the following format:

```
<LINK (extractionCommand [operator extractionCommand])
```

where *extractionCommand* is the member selection command to retrieve data from, and *operator* is either the AND or OR operator.

Note: You must select members from the same dimension, and all extraction command arguments must be enclosed in parentheses, as in the example above. NOT can only be associated with an extraction command, and does not apply to the entire expression.

You can use Boolean operators with member selection commands, such as UDA and wildcards. See the *Technical Reference* for a list of all valid extraction commands that can be used in conjunction with the LINK command.

Examples:

```
<LINK ((<IDESCENDANTS("100") AND <UDA(Product,Sweet)) OR
ONSAMELEVELAS "100"-10")
```

selects sweet products from the “100” subtree, plus all products on the same level as “100-10.”

```
<LINK ((<IDESCENDANTS("100") AND NOT <UDA (Product,Sweet)) OR
ONSAMELEVELAS "100"-10")
```

selects products that are not sweet from the “100” subtree, plus all products on the same level as “100-10.

See the *Technical Reference* for additional examples of narrowing member selection criteria.

Selecting Members by Using Substitution Variables

Substitution variables act as global placeholders for information that changes regularly; you set the substitution variables on the server through Essbase Administration Services, MaxL, or ESSCMD, and assign a value to each variable. You can then change the value at any time, reducing manual changes to a report script. You must have the role of at least Database Designer to set substitution variables.

For example, many reports are dependent on reporting periods; if you generate a report based on the current month, you must manually update the report script every month. With a substitution variable set on the server, such as CurMnth, you can change the assigned value each month to the appropriate time period. Analytic Services dynamically updates the information when you run the final report.

See "Using Substitution Variables" on page 133 for a comprehensive discussion of creating and changing substitution variables in the database outline. See the *Technical Reference* for information about the leading & character.

You can set substitution variables at the following levels:

- Server, providing access to the variable from all applications and databases on the server
- Application, providing access to the variable from all databases within the application
- Database, providing access to the specified database

➤ To use a substitution variable add the following to the report script:

The substitution variable must be accessible from the application and database against which you are running the report.

At the point in the script where you want to use the variable, use the following format:

```
&variableName
```

where *variableName* is the same as the substitution variable set on the server.

For example,

```
<ICHILDREN &CurQtr
```

becomes

```
<ICHILDREN Qtr1
```

Note: The variable name can be an alphanumeric combination whose maximum size is specified in Appendix A, "Limits." You cannot use spaces or punctuation in the variable name.

When you run the report script, Analytic Services replaces the variable name with the substitution value and that information is incorporated into the final report.

Selecting Members by Using Attributes

Using attributes, you can select and report on data based on one or more characteristics of base members. You can group and analyze members of base dimensions according to their attributes. You can also perform crosstab reporting based on two or more attributes. Using the <ATTRIBUTE command, you can select all the base dimension members associated with an attribute. For example, you can query the Sample Basic database on how many 12-ounce units of grape flavored juice and orange flavored juice were sold in New York during the first quarter.

➤ To select a member based on a specific attribute add the following to the report script:

At the point in the script where you want to select members based on a specific attribute, use the following format:

```
<ATTRIBUTE memberName
```

where *memberName* is the name of an attribute-dimension member; for example:

```
<ATTRIBUTE Bottle
```

returns all products packaged in bottles.

There can be cases where attribute dimensions have members with the same name. For example, the attribute dimension Ounces and the attribute dimension Age can each have a member named 24. To ensure that a query returns correct results, you must specify the full attribute-dimension member name. The following format returns all products that are packaged in 24 oz. units:

```
<ATTRIBUTE Ounces_24
```

Attribute types can be text, numeric, date, and Boolean. For a description of each attribute type, see “Understanding Attribute Types” on page 187.

Selecting Members by Attribute Association

You can select all the base dimension members associated with one or more attributes using the <WITHATTR command. For example, you can display all products associated with a member of the Pkg Type attribute dimension. At the point in the script where you want to select the members, enter the following syntax:

```
<WITHATTR (attributeDimensionName, "Operator", Value)
```

The following command returns all base dimension members that are associated with the attribute Small from the Population attribute dimension.

```
<WITHATTR (Population, "IN", Small)
```

The following command returns all base dimension members that are associated with the attribute 32 from the Ounces attribute dimension.

```
<WITHATTR (Ounces, "<", 32)
```

Note: The <WITHATTR command can be used within the LINK command to refine member selections, as illustrated in the following example:

```
<LINK ((<WITHATTR (Ounces, "<", 32) AND <WITHATTR ("Pkg Type",
"=", Can))
```

Selecting Members by Date

You can extract attributes data for a specific date, for a period before a specific date, or for a period after a specific date using the <TODATE command. For example, you can extract information on all products that were introduced on December 10, 1996, before December 10, 1996, or after December 10, 1996. The <TODATE command must be used within the <WITHATTR command. For example, the following format returns data on all products that were introduced *on* December 10, 1996.

```
<WITHATTR ("Intro Date", "=", <TODATE ("mm-dd-yyyy",
"12-10-1996")
```

The following format returns data on all products that were introduced *before* December 10, 1996.

```
<WITHATTR ("Intro Date", "<", <TODATE ("mm-dd-yyyy",
"12-10-1996")
```

The following format returns data on all products that were introduced *after* December 10, 1996.

```
<WITHATTR ("Intro Date", ">", <TODATE ("mm-dd-yyyy",
"12-10-1996")
```

Note: The types of date format supported are mm-dd-yyyy or dd-mm-yyyy. The date must be between January 1, 1970 and January 1, 2038 (inclusive).

Selecting Members by Using UDAs

A user-defined attribute (UDA) enables you to select and report on data based on a common characteristic. These attributes are particularly useful when performing member selections from an outline with an unbalanced hierarchy (a hierarchy where the members of a dimension do not have identical member levels). You can set UDAs on the server for characteristics such as color, size, gender, flavor, or any other common member characteristics. You must have Database Designer permissions to set UDAs on the server.

UDAs are different from attributes. UDAs are member labels that you create to extract data based on a particular characteristic, but you cannot use UDAs to group data, to perform crosstab reporting, or to retrieve data selectively. Hence, for data analysis, UDAs are not as powerful as attributes.

You can use the UDA command in conjunction with Boolean operators to refine report queries further. See "Selecting Members by Using Boolean Operators" on page 701 for examples of the UDA command being used with a Boolean operator.

See "Creating UDAs" on page 176 for information about creating and maintaining UDAs.

➤ To select members based on a UDA add the following to the report script:

At the point in the script where you want to select members based on the UDA, use the following format:

```
<UDA (dimensionName,"UDAstring")
```

where *dimensionName* is the dimension of the member that you select, and *UDAstring* is the UDA that is set on the server. The following example selects members of the Product dimension with the Sweet attribute.

```
<UDA (product,"Sweet")
```

When you run the report script, Analytic Services incorporates the UDA members into the final report.

Note: You must type the UDA string exactly as it is displayed in the database outline; you cannot create your own UDA string and incorporate it into the report script.

Selecting Members by Using Wildcards

You can use wildcards to select either members, generation, or level names in a report script. If you use member names, Analytic Services searches the member and all descendants of that member. If you specify a generation or level name, Analytic Services searches only members of that generation or level.

Using wildcards reduces the amount of member information needed for a script and simplifies script maintenance.

The following two types of wildcards are supported in Report Writer:

- Trailing asterisk (*) characters at the end of the string to search for common member properties
- Pattern-matching question mark (?) characters anywhere in the string to represent any single-character member property

➤ To select members using a trailing wildcard add the following to the report script:

At the point in the script where you want to select members using a trailing wildcard, use the following format:

```
<MATCH (memberName,"character*")
```

where *memberName* is the name of the member that you select, and *character* is the beginning character in the following member. Using the Sample Basic database,

```
<MATCH (Year,"J*")
```

returns Jan, Jun, and Jul.

➤ To select members using a pattern-matching wildcard add the following to the report script:

At the point in the script where you want to select members using a pattern-matching wildcard, use the following format:

```
<MATCH (memberName,"???characters")
```

where *memberName* is the name of the member to select, and *characters* are the characters in the following member. Using the Sample Basic database,

```
<MATCH (Product,"???-10")
```

returns 100-10, 200-10, 300-10, and 400-10.

Note: In the Sample Basic database example, three question marks are used to represent the variable three characters in the string. If only two question marks were used in the example, no matches are found. You can place question mark wildcards anywhere in the match string.

Selecting Members by Using Static Member Names

Static member names are unchanging member names, such as Sales and COGS, that you enter directly into the report script. Although they are easy to establish, static member name definitions can be difficult to maintain if the database outline changes. You must change the member name in every script that is used with that database.

In general, report scripts are easier to maintain if you use generation and level names, or member selection commands, rather than static member names whenever possible. For example, when you remove a product, such as Widgets, from the dimension, the member selection command <children Product works, but a specific list referencing Widgets as a static member name generates the following error message:

```
"Unknown Member [Widgets]."
```

Static members tend to be particularly difficult to maintain when dealing with distributed OLAP databases, where both source and target databases must be consistent.

Note: These user-defined, static member generation and level names are not displayed in Outline Editor.

The following report script uses a static member called ProductGroups to select all the Level 1, Product member names from the Sample Basic database.

```
<PAGE(Year)
<COLUMN(Product)
<ROW (Measures)
Qtr1
ProductGroups
Sales Profit
    !
```

The report script produces the following report:

```
                    Qtr1 Market Scenario
                100      200      300      400      Diet
            ======== ======== ======== ======== ========
Sales         25,048   26,627   23,997   20,148   25,731
    Profit     7,048    6,721    5,929    5,005    7,017
```

When you run a report that includes static member definitions, the report displays members in order of their definition in the report script by member name. Sort commands have no effect on static member definitions. See "Sorting Members" on page 712 for a discussion of the effects of sorting members.

Suppressing Shared Members

You can suppress the display of duplicate shared members when you extract data for a report. You can only suppress shared member display in conjunction with the following items:

- Generation names
- Level names
- DIMBOTTOM command
- OFSAMEGEN command
- ONSAMELEVELAS command

Suppressing shared members is useful when you want to eliminate unnecessary duplication of data within the report.

➤ To suppress shared members add the following to the report script:

At the point in the script from which you want to suppress a shared member, use

<SUPSHARE

This command suppresses the display of shared members for the duration of the report script. Use the <SUPSHAREOFF command to reset the display of shared members in the script.

Selecting Alias Names for Members

Aliases make reports easier to read and help the reader focus on the data values rather than the meanings of member names. You can display members in a report by their aliases. For example, you can display page, column, and row names, such as Diet Cola or Caffeine Free Cola, rather than the corresponding member names 100-20 and 100-30.

Task	Report Command
Display the alias set in the current alias table, without the member name.	OUTALT
Display the alias set in the current alias table, followed by the member name.	OUTALTMBR
Display the member name, followed by the alias set in the current alias table.	OUTMBRALT
Display the alias set in the current alias table, without the member name, for all members in the report.	OUTALTNAMES
Reset the default display of member names after using the OUTALTNAMES command.	OUTMBRNAMES
Include several alias tables within one report script.	OUTALTSELECT

For a complete listing of alias commands, see the *Technical Reference*.

To Display a Member Name and Alias

You can display members in a report as a combination of the member name and its alias. Combining the member name and alias lets you display more descriptive page, column, and row names, such as Diet Cola 100-20 or 100-30 Caffeine Free Cola.

To display a member name and alias, use the <OUTALTNAMES and <OUTALTMBR commands together in a report script, as illustrated in the following example:

```
<PAGE (Product, Measures)
<COLUMN (Scenario, Year)
{OUTALTNAMES}
<OUTALTMBR
Actual
<ICHILDREN Qtr1
<ROW (Market)
<IDESCENDANTS "300"
    !
```

The report script produces the following report:

```
                  Dark Cream 300-10 Measures Actual
                       Jan      Feb      Mar      Qtr1
                  ======== ======== ======== ========
Market                 800      864      880    2,544

                  Vanilla Cream 300-20 Measures Actual
                       Jan      Feb      Mar      Qtr1
                  ======== ======== ======== ========
Market                 220      231      239      690

                  Diet Cream 300-30 Measures Actual
                       Jan      Feb      Mar      Qtr1
                  ======== ======== ======== ========
Market                 897      902      896    2,695

                     Cream Soda 300 Measures Actual
                       Jan      Feb      Mar      Qtr1
                  ======== ======== ======== ========
Market               1,917    1,997    2,015    5,929
```

Sorting Members

When you sort the members you include in a report, be aware that sorting commands affect members differently, depending on whether they are referenced by member selection commands or by static member definitions. Report Writer commands sort members either by member name or data values.

Member selection commands such as <CHILDREN and <DESCENDANTS, select members in the order specified by the database outline. By default, a report that includes member selection commands displays members in their hierarchical database outline order. You can override this default by specifying a sort order with a sort command.

Because sort commands affect the order of the members selected by the member selection commands, they must precede any member selection commands to which they apply. If you specify a sort command, the sort order is preserved until another sort command overrides it.

Sort commands modify member selection commands, such as <CHILDREN and <DESCENDANTS. Sort commands do not perform any final sorting of rows during formatting. Be careful when you place a sort command in the report script that you do not start the sort too soon, and that you override it to turn it off, if necessary, before the next selection command.

Sort commands have no effect on static member definitions.

Task	Report Command
Sort all members alphabetically by the alias name of the member, if aliases are used in the report script.	SORTALTNAMES
Sort all following members in ascending order starting with the lowest generation and moving toward the highest generation.	SORTASC
Sort all following members in descending order starting with the highest generation and moving toward the lowest generation.	SORTDESC
Sort all following members according to the generation of the member in the database outline.	SORTGEN
Sort all following members according to the level of the member in the database outline.	SORTLEVEL

Task	Report Command
Sort all members alphabetically by member name.	SORTMBRNAMES
Disable all previous sorting commands so that members added to the report follow the normal hierarchical order based on the database outline.	SORTNONE

For a list of sorting commands syntax and descriptions, see the *Technical Reference*.

Restricting and Ordering Data Values

Several Report Writer commands let you perform conditional retrieval and data sorting in reports.

Task	Report Command
Specify the number of rows to return. These rows must contain the top values of a specific data column.	TOP
Specify the number of rows to return. These rows must contain the lowest values of a specific data column.	BOTTOM
Specify the conditions the columns of a data row must satisfy before the row is returned.	RESTRICT
Specify the ordering of the rows of a report, based on the data values of data columns.	ORDERBY

For the syntax, definitions, and detailed examples of these commands, see the *Technical Reference*.

Configurable variables are used during conditional retrievals. For information about setting the Report Writer configurable variables, see "Changing Buffer Size" on page 1244.

Understanding the Order of Operation

<RESTRICT, <ORDERBY, <TOP, and <BOTTOM can be displayed anywhere in the report script and in any order. When using these commands, place all global script formatting commands before a Page member or a Column member, or before a <PAGE command or <COLUMN command that expands into Page or Column members (for example, IDESCENDANTS, or ICHILDREN).

Analytic Services extracts data and applies restrictions and ordering in the following order:

1. Applies RESTRICT and any existing restrictive option such as SUPPMISSING, SUPZEROS, and SUPEMPTYROWS.
2. Applies TOP or BOTTOM, or both.
3. Applies ORDERBY.

Analytic Services then returns rows and displays output.

Using TOP, BOTTOM, and ORDERBY with Sorting Commands

<TOP, <BOTTOM, and <ORDERBY commands sort the output of a report by its data values. Analytic Services applies <TOP and <BOTTOM first, followed by <ORDERBY. If the report contains a sort command, such as <SORTMBRNAMES, which sorts members and not data, Analytic Services applies the sort command first, followed by <TOP and <BOTTOM, and then <ORDERBY. <ORDERBY is the final sort that takes place.

Using RESTRICT

The arguments of the <RESTRICT command let you specify qualifications for selecting rows. Analytic Services includes only qualified rows in the resulting report output.

<RESTRICT works only on the range of rows that you specify in a row member selection.

Analytic Services processes the restrictions from left to right, and does not allow grouping with parentheses in the list of arguments.

For example, the following example is *not* a valid syntax:

```
RESTRICT (... (@DATACOL(1) > 300 AND @DATACOL(2) < 600)...)
```

Only one <RESTRICT is allowed per report, as terminated by the **!** command. If a report script contains more than one report, each <RESTRICT overwrites the one in the previous report, as illustrated in the following example:

```
RESTRICT (@DATACOL(1) > @DATACOL(2) AND 800 < @DATACOL(3)
OR @DATACOL(4) <> #MISSING)
```

This <RESTRICT command is equivalent in operation to the following syntax:

```
RESTRICT (((@DATACOL(1) > @DATACOL(2)) AND (800<@DATACOL(3))) OR
(@DATACOL(4) <> #MISSING))
```

Using ORDERBY

The <ORDERBY command orders the output rows according to the data values in the specified columns. Using an optional direction argument to the ORDERBY command, you can specify either an ascending order using the ASC flag, or descending order using the DESC flag.You can specify different sorting directions in different columns of the same report. If no direction argument is used, ascending (ASC) is the default order.

To determine the set of rows to be ordered, specify the row grouping dimension in the command. The default row grouping is the innermost row dimension.

Only one <ORDERBY is allowed per report, as terminated by the ! command. If a report script contains more than one report, each <ORDERBY overwrites the one in the previous report.

Using ORDERBY with Formatting Commands

Follow these guidelines when using the <ORDERBY command in a report script:

- Avoid using row formatting commands when you are using <ORDERBY in a report. Formatting commands scheduled for a given point in the report may show up unexpectedly because <ORDERBY shifted the row that contained the member with formatting.
- In general, avoid using row formatting commands, and place overall script formatting before column members or commands that expand the column members (such as “ICHILDREN column dimension, <column ..., column member”).

Using TOP and BOTTOM

The <TOP and <BOTTOM commands specify the qualified number of rows with the highest or lowest column values, respectively, within a row group to be returned in a report. If the row group member is not specified, the innermost row group dimension is the default row group.

You can use <TOP and <BOTTOM together in the same report, but only one <TOP and one <BOTTOM is allowed per report. In this case, the two commands should have the same data column as their argument in order to prevent confusion. The result of the <TOP and <BOTTOM command is sorted by the value of the data column specified in the command in descending order.

<TOP and <BOTTOM work only on the range of rows specified in row member selection.

Note: If <TOP or <BOTTOM occurs with <ORDERBY, the ordering column of the <ORDERBY does not have to be the same as the data column of the <TOP or the <BOTTOM.

If any combination of the <ORDERBY, <TOP, or <BOTTOM commands exist together in a report script, the row group member (<*rowGroupMember*>) should be the same. This restriction removes any confusion about the sorting and ordering of rows within a row group.

CAUTION: Analytic Services discards rows that contain #MISSING values in their sorting column from the set of extracted data rows before the applying the TOP or BOTTOM sort.

For example, this command returns two rows with the highest data values in col2 (Actual, Qtr2) per row group:

```
1- TOP (2, @DATACOL(2))
```

When you run this command against the Sample Basic database, the row grouping is Product, which implies that for Florida, the report returns 100-10 and 100-30 product rows, and for Maine, the report returns 100-10, 100-40 product rows, and so on.

		Actual		Budget	
		Qtr1	Qtr2	Qtr1	Qtr2
Florida	100-10	570	670	570	650
	100-20	235	345	321	432
	100-30	655	555	455	865
	100-40	342	342	432	234
Maine	100-10	600	800	800	750
	100-20	734	334	734	534
	100-30	324	321	235	278
	100-40	432	342	289	310
New York	100-10	1010	1210	1110	910
	100-20	960	760	650	870
	100-30	324	550	432	321
	100-40	880	980	880	1080
	100-50	#MI	#MI	#MI	#MI

This example returns rows with the highest data values in col2 (Actual, Qtr2) per report, because the row grouping is the “market.”

```
2- TOP("market", 3, @DATACOL(2))
```

This command returns the following rows:

New York	100-10	1010	1210	1110	910
	100-40	880	980	880	1080
Maine	100-10	600	800	800	750

This example returns two rows with the lowest data values in col2 (Actual, Qtr2) per row group.

```
3- BOTTOM ("market", 2, @DATACOL(2))
```

This command returns the following rows:

```
Maine       100-20      734       334       734       534
            100-30      324       321       235       278
```

Note: <TOP and <BOTTOM put an upper limit on the number of (qualified) rows returned after all restrictions are applied. This upper limit is equal to the number of rows in the <TOP plus the number of rows in the <BOTTOM commands.

Understanding How Other Report Configurations Affect TOP and BOTTOM

When using the <TOP and <BOTTOM commands, be aware that some other commands affect their operation. In particular, Analytic Services treats the <SUPPMISSING, <SUPZEROS, and <SUPEMPTYROWS options as restrictions and applies them to the extracted rows along with the <RESTRICT command restrictions. Analytic Services applies these optional restrictions to the data rows before processing the <TOP or <BOTTOM commands, and before applying an <ORDERBY command.

Using TOP and BOTTOM with Formatting Commands

Whenever a formatting command occurs in a report, it is appended to the member that follows it. For example, in this sequence, {UCOLUMNS}, the underline columns command is appended internally to the member that comes next. In Script 1, it is appended to the row member that can be described as “first child of market, assuming Florida.”

```
SCRIPT 1                    SCRIPT 2
....                        ....
< ROW Market                {UCOL}
{UCOL }                     < Florida    (row member)
<ICHILDREN Market
< TOP ....                  < BOTTOM ....
```

Script 2, appends {UCOLUMNS} to the row member Florida. Analytic Services executes {UCOLUMNS} whenever it encounters a row that has row member Florida. If the TOP or BOTTOM command returns a row that does not contain Florida, the formatting commands appended to the rows are never executed.

Therefore, it is a good idea to place all general formatting commands before a <COLUMN command, or a command that expands into column members to guarantee that the formatting is executed. However, you should not use formatting commands that work on rows, because these rows may never be picked up by the <TOP or <BOTTOM command. Also avoid using <SAVEROW and <CALCULATE ROW with the <TOP and <BOTTOM commands.

Converting Data to a Different Currency

If the database has a currency partition, you can calculate currency conversions in report scripts. Use the <CURRENCY command to set the output currency and currency type. Use the <CURHEADING command to display the currency conversion heading.

Note: Currency conversion is not supported across transparent partitions.

For information about creating a currency conversion application, see "Building Currency Conversion Applications and Performing Conversions" on page 222.

For the syntax and definitions of Report Writer commands, see the *Technical Reference*.

Generating Reports Using the C, Visual Basic, and Grid APIs

Use the following table to determine the report API calls that you can make:

Task	C API Function	Visual Basic API Function	C Grid API Function
Start sending a report specification to the active database.	ESSBEGINREPORT	ESBBEGINREPORT	ESSGBEGINREPORT
Mark the end of a report specification being sent to the active database.	ESSENDREPORT	ESBENDREPORT	N/A
Send a report specification to the active database as a single string.	ESSREPORT	ESBREPORT	N/A
Send a report specification to the active database from a file.	ESSREPORTFILE	ESBREPORTFILE	ESSGREPORTFILE

See the *API Reference* for syntax and descriptions of these API functions.

Chapter 33

Mining an Analytic Services Database

Data mining is the process of searching through an Analytic Services database for hidden relationships and patterns in a large amount of data. Using traditional query tools, you can answer many standard questions about your business, such as "How profitable were root beer sales in New York in July?" Data mining helps you to search through large amounts of data and come up with new questions, such as "What will root beer sales look like in August?"

This chapter describes data mining and explains how to mine Analytic Services databases.

Understanding Data Mining

Data mining tools can sift through data to come up with hidden relationships and patterns. You may find that people who bought root beer in July also bought ice cream in July. Then you can use this knowledge to create an advertising campaign around root beer floats.

You can use data mining to tell you things about existing data, as in the root beer example. Such data mining is called descriptive. You can also use data mining to forecast future results based on past performance. For example, you can forecast sales for next year based on sales for this year. Such data mining is called predictive.

Essbase Analytic ServicesData Mining Framework

Data Mining Framework is a collection of features that enables the performance of data mining on Analytic Services databases. Data Mining Framework is licensed separately from Analytic Services.

Note: Data Mining Framework does not currently operate on relational data, that is, hybrid analysis data. Data mining is also not supported currently for Unicode-mode applications.

The process of mining an Analytic Services database includes the following tasks:

- Specify a build task.

  You implement a build task through a task template. The Mining Wizard, available with Administration Services, provides a build task template and steps you through the process of specifying a build task. You can also use MaxL statements and template files provided with Data Mining Framework.

  In the build task, you specify:

  – The algorithm to use.

  – The database to mine and a representative set of data.

  – Parameters that determine how the algorithm is applied.

- Execute the build task to build, or train, a model.

  When you execute a build task, the specified algorithm is applied against the the specified set of data to generate a data mining model. During the training process, the algorithm discovers and describes the patterns and relationships in the data that can be used for prediction. Later, the trained model can use these patterns and relationships to generate new information from a different, but similarly structured, set of data.

  See “Training the Model” on page 726.

- Specify a test task to test the model. This is an optional task to verify the accuracy of the model you have built.

  You implement a test task through a task template using the Mining Wizard or MaxL statements.

In the test task, you specify:

- The model to use. This is a model that you have built.
- A database and set of data to mine. Specify a set of data with known results or correlations.

- Execute the test task to generate test results.

  After training the model on one set of data, you execute the model against a different set of data with known results or correlations. You can compare the results generated by the model with the known results to determine the accuracy of the model.

  See "Testing the Model" on page 727.

- Specify an apply task.

  You implement an apply task through a task template using the Mining Wizard or MaxL statements.

  In the apply task, you specify:

  - The model to use. This is a model that you have built.
  - A database and set of data to mine

- Execute the apply task to generate result records.

  When you execute an apply task, the model is executed against the specified set of data to generate result records. Data Mining Framework writes results back to the Analytic Services cube.

  See "Applying the Model" on page 727.

- Query the result records.

  Query the result records to view the data mining results.

The following sections describe the data mining process in more detail.

Creating Data Mining Models

The first step in building a data mining model is to understand the business situation or problem and choose the appropriate algorithm to use for analysis or prediction. Algorithms determine how you search through data.

Data Mining Framework provides a set of powerful algorithms that can be used for a broad range of business applications. See "Built-in Algorithms" on page 728 for a description of the available algorithms. The description of each algorithm provides information about the type of problem that it is best suited for.

Note: Data Mining Framework also enables you to easily register and use new algorithms created by Hyperion or third-party vendors.

For example, suppose you want to determine how sales of televisions, DVDs, and VCRs in the East region correspond to sales of cameras in the same region. You can use the regression algorithm to model sales of cameras as a function of sales of TVs, VCRs, and DVDs.

The regression algorithm predicts the value of a dependent (target) variable based on the value of one or more independent (predictor) variables.

Using an Algorithm

All data mining algorithms require training, or learning. Training is the process of executing an algorithm against a representative set of data to discover and describe the patterns and relationships in the data that can be used for prediction. Once a model has been trained against a representative set of data, it can then be applied to a wider set of data to derive useful information.

Learning can be either supervised or unsupervised. Supervised learning discovers patterns and relationships in the data by comparing the resulting data to a set of known data. Unsupervised learning discovers patterns and relationships in the data without comparing the data to a known set of data.

The algorithm vendor determines the specific information that you must provide to use the algorithm. The regression algorithm employs supervised learning. Therefore, the model requires both input data and output data.

To use an algorithm, you need to enter its settings, parameters and accessors.

Settings are actually determined by the Data Mining Framework, and as such are the same for all algorithms, but they do influence the operation of the algorithm. For example, the framework provides a setting to define how missing values are treated by the algorithm.

Parameters are specific to each algorithm and determine how the algorithm operates on the data. For example, the clustering algorithm provides a parameter to specify the maximum number of clusters to return.

Accessors specify the input and output data for the algorithm. In a build task, supervised algorithms such as regression have two accessors, a predictor to define the independent or input data, and a target to define the dependent or expected output data. Unsupervised algorithms, such as clustering, have a predictor accessor only.

Test and apply tasks generally have both predictor accessors to define the input and target accessors to define the output.

Accessors consist of domains, which are typically MaxL DML expressions that define components of the accessor. For example, the predictor accessor for the regression algorithm contains the following domains:

- `Predictor.Predictor`

  Defines the member or member set combination that determines the predictor domain.

- `Predictor.Sequence`

  Defines the sequence to be traversed for the predictor variable. Sequence is generally a time dimension.

- `Predictor.External`

  Defines the scope of the predictor. It is optional.

- `Predictor.Anchor`

  Defines restrictions from additional dimensions. It is optional.

The target accessor has the same set of domains as the predictor. You write MaxL DML expressions to define the predictor and target accessors.

For example, consider this sample data mining problem:

Given the number of TVs, DVDs, and VCRs sold during a particular period, in the East region, how many cameras were sold in the same period in the East? Restrict sales data to prior year actual sales.

Using the regression algorithm, the predictor and target accessors to define the model for this problem are as follows:

```
Predictor.Predictor      {[Television], [DVD], [VCR]}
Predictor.Sequence       {[Jan 1].Level.Members}
Predictor.External       {[East].Children}
Predictor.Anchor         {([2001], [Actual], [Sales])}
Target.Target            {[Camera]}
Target.Sequence          {[Jan 1].Level.Members}
Target.External          {[East].Children}
Target.Anchor            {([2001], [Actual], [Sales])}
```

Note: In this example, the target accessor is the same with regard to all the predictor attributes except the target domain (`{[Camera]}`). However, the domain expressions for different accessors are not required to be the same. The only requirement is that a predictor component (for example `predictor.sequence`) and the corresponding target component (`target.sequence`) must be the same size.

For each city in the East (`{[East].Children}`), the algorithm models camera sales as a function of TV, DVD, and VCR sales. The Data Mining Framework creates, under the same name, a family of results, or models; a separate result for each city in the East.

Training the Model

The final step of specifying a build task is to execute the algorithm against the data specified by the accessors to build or train the model. During the training process, the algorithm discovers and describes the patterns and relationships in the data that can be used for prediction.

Internally, the algorithm represents the patterns and relationships it has discovered as a set of mathematical coefficients. Later, the trained model can use these patterns and relationships to generate new information from a different, but similarly structured, set of data.

Note: If you cancel a data mining model while you are training it, the transaction is rolled back.

Applying the Model

After the model is trained, it is ready to use on a new set of data. To apply a model to a new set of data, you specify an apply task. In the apply task you specify a build model you trained and a set of accessors. Generally, the values are the same for the predictor and sequence domains for a build task and its related apply task. You change the external or anchor domain to apply the model to a different set of data. For example, you could change the external domain to specify a different region or country. Or you could use the anchor domain to specify a different year.

In the apply task, the target result data is not known, but is to be predicted by the model.

The apply task applies the model coefficients it generated to the new set of data and generates a set of output data. The Data Mining Framework writes the result data back to the Analytic Services cube. The apply task generates a result record that you can use to query the result data.

Testing the Model

Data mining models are built using known data to train the algorithm so it can be applied to a similar data set. To test a model, you create a test task. In the test task, you specify a model you have trained and a set of accessors. In addition to the predictor and target accessors, you specify test accessors that reference a known set of results.

The test task compares the results of the trained model to the set of known results you specify. The test task determines if the results match within a specified range of expected error. If the results do not match, you can do any of the following:

- Verify the homogeneity of the known data. If the structure of the known data does not match the structure of the test data, the results will not match.
- Verify the integrity of the known data. Corrupt or incomplete data can cause the test model to fail.
- Verify the integrity of the test input data.
- Consider changing the stringency of the settings. At very least, a less stringent setting that returns a positive test gives you an idea of how closely the trained model compares to known data.

See “Creating or Modifying a Test Task” in *Essbase Administration Services Online Help* for information about creating a test task.

Viewing Data Mining Results

Data Mining Framework writes mining results back to the Analytic Services cube. Data Mining Framework creates a result record, in XML format, that contains accessors that specify the location of the result data in the cube.

You can view data mining results through the Data Mining node in Administration Services or by using MaxL statements.

Preparing for Data Mining

The one essential prerequisite for performing data mining is that you understand your data and the problem you are trying to solve. Data mining is a powerful tool and can yield new insights. However, if you already have a strong hunch about your data, then data mining can be particularly useful in confirming or denying your hunch, and giving you some additional insights and directions to follow.

Before you mine an Analytic Services database, make sure that the database is loaded and calculated.

Built-in Algorithms

Hyperion supplies the following basic algorithms:

- Regression. Identifies dependencies between a specific value and other values. For example, multilinear regression can determine how the amount of money spent on advertising and payroll affects sales values.
- Clustering. Arranges items into groups with similar patterns. You use the clustering algorithm for unsupervised classification. The algorithm examines data and determines itself how to split the data into groups, or clusters, based on specific properties of the data. The input required to build the model consists of a collection of vectors with numeric coordinates. The algorithm organizes these vectors into clusters based on their proximity to each other. The basic assumption is that the clusters are relatively smaller than the distance between them, and, therefore, can be effectively represented by their respective centers. Hence the model consists of coordinates of center vectors.

  Sequential runs on the same training set may produce slightly different results due to the stochastic nature of the method. You specify the number of clusters to generate, but it is possible the algorithm will find fewer clusters than requested.

Clusters can provide useful information about market segmentation and can be used with other predictive tools. For example, clusters can determine the kinds of users most likely to respond to an advertising campaign and then target just those users.

- Neural network. Generalizes and learns from data. For example, neural networks can be used to predict financial results.

  You can use the neural net algorithm for both prediction and classification. This algorithm is much more powerful and flexible than linear regression. For example, you can specify multiple targets as well multiple predictors.

  On the other hand, the model generated by the neural net algorithm is not as easy to interpret as that from linear regression.

  One use of neural nets is binary classification. A series of inputs (predictors) produces a set of results (targets) normalized to values between zero and one. For example, a set of behaviors results in values between 0 and 1, with 1 being risky and 0 being risk free. Values in between require interpretation; for example, 0.4 is the high end of safe and 0.6 is the low end of risky.

- Decision tree. Determines simple rules for making decisions. The algorithm results are the answers to a series of yes and no questions. A yes answer leads to one part of the tree and a no answer to another part of the tree. The end result is a yes or no answer. Decision trees are used for classification and prediction. For example, a decision tree can tell you to suggest ice cream to a particular customer because that customer is more likely to buy ice cream with root beer.

  Use the decision tree algorithm to organize a collection of data belonging to several different classes or types. In the build phase, you specify a set of data vectors and provide the class of each. In the apply phase, you provide a set of previously unknown vectors and the algorithm deduces their classes from the model.

  The algorithm constructs a series of simple tests or predicates to create a tree structure. To determine the class of a data vector, the algorithm takes the input data and traverses the tree from the root to the leaves performing a test at each branch.

- Association Rules. Discovers rules in a series of events. The typical application for this algorithm is market basket analysis: people who buy particular items also buy which other items. For example, the result of a market basket analysis might be that men who buy beer also buy diapers.

You define support and confidence parameters for the algorithm. The algorithm selects sufficiently frequent subsets selected from a predefined set of items. On input it reads a sequence of item sets, and looks for an item set (or its subset), whose frequency in the whole sequence is greater than support level. Such item sets are broken into antecedent-consequent pairs, called rules. Rule confidence is the ratio of its item set frequency to the antecedent frequency in all the item sets. Rules with confidence greater than the given confidence level are added to the list of "confident" rules.

Although the algorithm uses logical shortcuts during computations, thus avoiding the need to consider all the combinations of the item sets, whose number can be practically infinite, the speed with which the algorithm executes depends on the number of attributes to consider and the frequency with which they occur.

- Naive Bayes. Predicts class membership probabilities. Naive Bayes is a light-weight classification algorithm. It is fast, takes small memory and in a good number of applications behaves quite satisfactory, so you can use it first before going to the decision tree or fully fledged clustering schemes.

  The algorithm treats all the attributes of the case vector as if they were independent of each other. It uses a training sequence of vectors and the theoretical definition of the conditional probability to calculate the probabilities or likelihoods that an attribute with a certain value belongs to a case with a certain class. The model stores these probabilities. In the apply mode the case attributes are used to calculate the likelihood of the case for each class. Then a class with the maximal likelihood is assigned to the case.

Accessing Data Mining Functionality

Data Mining Framework is supported by the MaxL and Administration Services interfaces.

MaxL provides a set of statements explicitly for data mining. With MaxL you can perform all data mining functions, including creating, training, testing, and applying a data mining model.

Sample model templates for each of the algorithms are available through the Administration Services interface. A model template provides an outline of the accessors needed for that algorithm that you can fill in. It also sets some parameters required by the algorithm.

Administration Services enables you to manage data mining models, templates, transformations, and results. It provides a mining wizard that steps you through the process of creating and training a build model, and creating and applying apply and test models. See “Mining an Analytic Services Database” in *Essbase Administration Services Online Help*.

Creating New Algorithms

You can create your own algorithms using Java and register them with Data Mining Framework. In order to be recognized by Data Mining Framework, an algorithm must implement certain interfaces and have a specific signature. These requirements are described in detail in the *Algorithm Vendor’s Guide* shipped as part of the Data Mining Framework SDK.

After a new algorithm is registered, it appears in the list of supplied algorithms and you can use the new algorithm to create build and apply tasks. Data Mining Framework reads the instructions for each parameter in the algorithm from the algorithm signature. The instructions appear in the Build and Apply Wizards in the panels where the user sets the algorithm parameters, just like the instructions for the supplied algorithms.

Chapter

34 Copying Data Subsets and Exporting Data to Other Programs

You can move data between Analytic Services databases or to another program by extracting an output file of the data you want to move. For example, you can copy a subset of an Analytic Services database from an Analytic Server to Personal Essbase.

In order to meet the import format specifications of most other programs, use the Report Writer to create a text file.

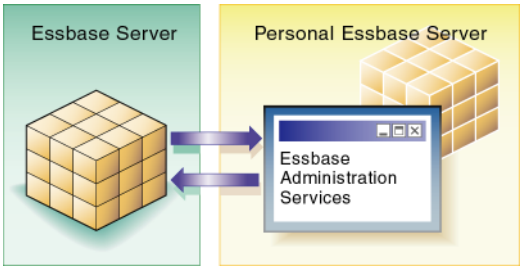
This chapter contains information about the following topics:

Copying a Database Subset

You can install both the Analytic Server and client on a Windows NT or Windows 2000 workstation using Personal Essbase. Personal Essbase is a one-port license and has its own license number. For information about installing and configuring Personal Essbase on a computer, see the *Essbase Analytic Services Installation Guide*.

Once you have installed Personal Essbase, you can copy the outline file (*dbname*.otl) and a data subset from the Analytic Server and load them into Personal Essbase. The Personal Essbase server does not communicate with the Analytic Server.

Figure 207: Analytic Services and Personal Essbase Interaction



Note: Do not create more than one application and two databases on the Personal Essbase server.

Process for Creating a Database Subset

This section describes the process for copying a database subset to Personal Essbase.

1. On the Personal Essbase server, create a new application and database to contain the database subset.

   See "Creating a New Application and Database" on page 735.

2. Copy the outline file (for example, `source_dbname.otl`) from the source database to the new database on Personal Essbase.

   You may need to rename the outline file to match the name of the Personal Essbase database (for example, `target_dbname.otl`), overwriting the existing target database outline file.

   See "Copying the Outline File from the Source Database" on page 735.

3. Create an output file (for example, an plain text file) containing the required data subset.

   See “Creating an Output File Containing the Required Data Subset” on page 737.

4. Load the output file into the new database that you have created.

   For instructions, see “Loading the Output File Into the New Database” on page 739.

If required, you can repeat steps 3 and 4 to create an output file from the database on the Personal Essbase server and load the data back into the main Analytic Services database on a different computer.

The example in the following sections is based on the Sample Basic database. The data subset in the example is the Actual, Measures data for the West market. The example copies the data subset to a Personal Essbase server and the West Westmkts database.

Creating a New Application and Database

Create a new application and database on the Personal Essbase server. You will copy the required subset of data into this new database. You can give this application and database any name.

➤ To create the application and database, see “Creating Applications” and “Creating Databases” in the *Essbase Administration Services Online Help*.

Ensure that the new, empty database is not running.

➤ To stop a database, see “Stopping Databases” in the *Essbase Administration Services Online Help*.

Copying the Outline File from the Source Database

Copy the outline file (`.otl`) of the source database to the new database that you have created. In this example, you copy the `basic.otl` outline file from the Sample Basic database and rename it to `wesmkts.otl` on Personal Essbase.

How you copy the outline file depends on whether you can connect to the source Analytic Services database from the Personal Essbase computer.

- If you *can* connect, use any of the following methods to copy the outline:

Tool	Topic	Location
Administration Services	Copying Outlines	*Essbase Administration Services Online Help*
MaxL	**create database**	*Technical Reference*
ESSCMD	COPYDB	*Technical Reference*

- If you *cannot* connect (for example, if the Personal Essbase computer is a laptop computer that has no network connection), use the operating system to copy the outline file.

  1. Use the operating system to copy the source outline file; for example, copy `basic.otl` to `westmkts.otl`.
  2. Give the copied outline exactly the same name as the new database.
  3. Save the outline file in the `\ARBORPATH\App\appname\dbname` directory on the Personal Essbase computer, where *ARBORPATH* is the directory in which you installed Analytic Services, and *appname* and *dbname* are the new application and database that you have created.

     For example, copy `basic.otl` to a disk, renaming it to `westmkts.otl`. Then copy `westmkts.otl` from the disk to `c:\essbase\app\west\westmkts\westmkts.otl` on the Personal Essbase computer. It is safe to overwrite the existing, empty `westmkts.otl` file.

     Note: Ensure that the new outline file overwrites the existing, empty outline file, which Analytic Services created automatically when you created the new application and database.

  4. Stop and then restart the new database.

     See "Starting Databases" and "Stopping Databases" in the *Essbase Administration Services Online Help*.

You now have a copy of the database outline on the Personal Essbase server.

Creating an Output File Containing the Required Data Subset

Create an output file that contains the required data subset. The output file can be a text file or a spreadsheet file. Use either of the following methods to create a data subset.

➤ To create an output file using the Report Writer, see "Executing Report Scripts" in the *Essbase Administration Services Online Help*.

➤ To create an output file using the Retrieval Wizard, see the *Essbase Spreadsheet Add-in User's Guide*.

The following example shows how to create a subset of the Westmkts database.

➤ To create a text file that contains the required data subset, follow these steps:

1. Select the source database. For example, select West Westmkts.

   See "Navigating and Selecting Objects" in the *Essbase Administration Services Online Help*.

   - If you *can* connect to the main Analytic Services database from the Personal Essbase computer, you can select the source database from the Personal Essbase computer.
   - If you *cannot* connect, use a different computer from the Personal Essbase computer to select the source database.

2. Create a new report.

   See "Creating Scripts" in the *Essbase Administration Services Online Help*.

3. Write a report script that selects the required data subset. For fundamental information on writing report scripts, see Chapter 31, "Understanding Report Script Basics."

   For example, the following report script selects the Actual, Measures data for the West market from Sample Basic:

   Figure 208: Sample Basic Report Script

```
{TABDELIMT}
<QUOTEMBRNAMES
Actual
```

```
<IDESC West
<IDESC Measures
```

- Use TABDELIMIT to place tab stops between data, instead of spaces to ensure that no member names or data values are truncated.
- Use QUOTEMBRNAMES to place quotation marks ("") around member names that contain blank spaces. Analytic Services then recognizes the member names when it loads the data.

4. Execute the report script.

   See "Executing Report Scripts" in the *Essbase Administration Services Online Help*.

5. Save the report script with a `.txt` extension; for example, `westout.txt`.

   To load the data, the output file needs to be in the `\ARBORPATH\App\appname\dbname` directory on the Personal Essbase server, where *ARBORPATH* is the directory in which you installed Analytic Services, and *appname* and *dbname* are the new application and database directories that you have created.

   If you are using the Personal Essbase computer, you can save the output file directly into the `\ARBORPATH\app\appname\dbname` directory, for example, `c:\essbase\app\west\westmkts\westout.txt`.

   If you are *not* using the Personal Essbase computer, save the output file anywhere on the current computer. By default, Analytic Services saves the file on the Analytic Services client computer, and not on the server. When you run the report, use the operating system to copy the file to the `\ARBORPATH\App\appname\dbname` directory on the Personal Essbase server. For example, use a disk to copy the file.

   If you are *not* using the Personal Essbase computer, remember to download and copy the file from the Analytic Server client directory to the `\ARBORPATH\app\appname\dbname` directory on the Personal Essbase server. For example, copy the output file to `c:\essbase\app\west\westmkts\westout.txt`.

You are now ready to load the text file into the new database.

Loading the Output File Into the New Database

Load the output file into the new database on the Personal Essbase computer.

To load a file into a database, see "Performing a Data Load or Dimension Build" in the *Essbase Administration Services Online Help*.

The following example illustrates how to load data into the Westmkts database:

1. Select the new database. For example, select `Westmkts`.
2. Start the data load using the text file you have just created, for example, westout.

   Note: If westout is not displayed, check that you gave it a `.txt` extension and placed it in the `\`*`ARBORPATH`*`\App\West\Westmkts` directory. See "Creating an Output File Containing the Required Data Subset" on page 737 for instructions for naming an output file.

For detailed information on loading data and any errors that may occur, see Chapter 19, "Performing and Debugging Data Loads or Dimension Builds."

You can now view the data on the Personal Essbase computer. You might need to recalculate the database subset. Because you are viewing a subset of the database, a percentage of the data values will be #MISSING.

If required, you can copy report scripts and other object files to the Personal Essbase computer to use with the database subset you have created.

Exporting Data Using Report Scripts

You can use report scripts to export Analytic Services data to other programs in text format. Report Writer enables you to create text files that meet the import format specifications of most other programs.

For information about exporting databases using other methods, see "Exporting Data" on page 743.

Before you can import data into some programs, you must separate, or delimit, the data with specific characters.

If you plan to import Analytic Services data into a program that requires special delimiters, use the MASK command.

Note: You cannot export data generated by Dynamic Calc members. Because attributes are Dynamic Calc members, you cannot export data generated by attributes.

When you export data to a program that uses a two-dimensional, fixed-field format, you do not need to specify page or column dimensions. To create a two-dimensional report, you can specify every dimension as a row dimension. Use the ROWREPEAT command to add the name of each member specified to each row (rather than the default, nested style). The following script example and report illustrate this situation for a five-dimensional database:

```
<ROW (Year, Measures, Product, Market, Scenario)
{ROWREPEAT}
<ICHILDREN Year
Sales
<ICHILDREN "400"
East
Budget
    !
```

This script produces the following report:

```
Qtr1          Sales         400-10        East       Budget       900
Qtr1          Sales         400-20        East       Budget     1,100
Qtr1          Sales         400-30        East       Budget       800
Qtr1          Sales           400         East       Budget     2,800
Qtr2          Sales         400-10        East       Budget     1,100
Qtr2          Sales         400-20        East       Budget     1,200
Qtr2          Sales         400-30        East       Budget       900
Qtr2          Sales           400         East       Budget     3,200
Qtr3          Sales         400-10        East       Budget     1,200
Qtr3          Sales         400-20        East       Budget     1,100
Qtr3          Sales         400-30        East       Budget       900
Qtr3          Sales           400         East       Budget     3,200
Qtr4          Sales         400-10        East       Budget     1,000
Qtr4          Sales         400-20        East       Budget     1,200
Qtr4          Sales         400-30        East       Budget       600
Qtr4          Sales           400         East       Budget     2,800
  Year        Sales         400-10        East       Budget     4,200
  Year        Sales         400-20        East       Budget     4,600
  Year        Sales         400-30        East       Budget     3,200
  Year        Sales           400         East       Budget    12,000
```

If you want to create a two-dimensional report that contains only bottom-level (level 0) data, use CHILDREN or DIMBOTTOM to select level 0 members.

- To list only level 0 data for specific members, use the CHILDREN command with the level 1 member as a parameter above the data you want to print.
- To list all level 0 data for the dimension to which a given member belongs, use the DIMBOTTOM command with any member in the dimension that contains the data you want to print.

For example, the following script uses the CHILDREN command to select the children of Qtr1, which is a level 1 member, and the DIMBOTTOM command to select all level 0 data in the Product dimension.

```
<ROW (Year, Measures, Product, Market, Scenario)
{ROWREPEAT}
{DECIMAL 2}
<CHILDREN Qtr1
Sales
<DIMBOTTOM Product
East
Budget
     !
```

This script produces the following report:

```
Jan      Sales    100-10     East       Budget        1,600.00
Jan      Sales    100-20     East       Budget          400.00
Jan      Sales    100-30     East       Budget          200.00
Jan      Sales    200-10     East       Budget          300.00
Jan      Sales    200-20     East       Budget          200.00
Jan      Sales    200-30     East       Budget        #Missing
Jan      Sales    200-40     East       Budget          700.00
Jan      Sales    300-10     East       Budget        #Missing
Jan      Sales    300-20     East       Budget          400.00
Jan      Sales    300-30     East       Budget          300.00
Jan      Sales    400-10     East       Budget          300.00
Jan      Sales    400-20     East       Budget          400.00
Jan      Sales    400-30     East       Budget          200.00
Feb      Sales    100-10     East       Budget        1,400.00
Feb      Sales    100-20     East       Budget          300.00
Feb      Sales    100-30     East       Budget          300.00
Feb      Sales    200-10     East       Budget          400.00
Feb      Sales    200-20     East       Budget          200.00
Feb      Sales    200-30     East       Budget        #Missing
Feb      Sales    200-40     East       Budget          700.00
Feb      Sales    300-10     East       Budget        #Missing
Feb      Sales    300-20     East       Budget          400.00
Feb      Sales    300-30     East       Budget          300.00
Feb      Sales    400-10     East       Budget          300.00
Feb      Sales    400-20     East       Budget          300.00
Feb      Sales    400-30     East       Budget          300.00
Mar      Sales    100-10     East       Budget        1,600.00
Mar      Sales    100-20     East       Budget          300.00
Mar      Sales    100-30     East       Budget          400.00
Mar      Sales    200-10     East       Budget          400.00
Mar      Sales    200-20     East       Budget          200.00
Mar      Sales    200-30     East       Budget        #Missing
Mar      Sales    200-40     East       Budget          600.00
Mar      Sales    300-10     East       Budget        #Missing
Mar      Sales    300-20     East       Budget          400.00
Mar      Sales    300-30     East       Budget          300.00
Mar      Sales    400-10     East       Budget          300.00
Mar      Sales    400-20     East       Budget          400.00
Mar      Sales    400-30     East       Budget          300.00
```

For an additional example of formatting for data export, see “Sample 12 on the Examples of Report Scripts” page in the “Report Writer Commands” section of the *Technical Reference*.

Importing Data Into Other Databases

Before you import data into some programs, you must delimit the data with specific characters. If you plan to import Analytic Services data into a program that requires special delimiters, use the MASK command.

Exporting Data

➤ To export data from a database, use any of the following methods:

Tool	Topic	Location
Administration Services	Exporting Data	*Essbase Administration Services Online Help*
MaxL	**export data**	*Technical Reference*
ESSCMD	EXPORT	*Technical Reference*

Note: Export files from databases in Unicode-mode applications are in UTF-8 encoding.

For more information about exporting data, see “Export Backups” on page 1082.

Chapter

35

Writing MDX Queries

This chapter describes MDX, the data manipulation language for Analytic Services.

MDX is a joint specification of the XML for Analysis founding members. For more information about XML for Analysis, please visit http://www.xmla.org. For more details on the syntax and grammar of MDX, refer to the MaxL section of the *Technical Reference*.

The goal of this chapter is to familiarize you with MDX and develop simple queries based on the Sample Basic database.

To complete the exercises in this chapter, use the MaxL Shell. Before continuing, please start Analytic Services, and log in to the MaxL Shell. Additionally, be prepared to use a text editor to create the sample queries as presented in this chapter.

Note: You can use the MDX Script Editor in Administration Services Console instead of the MaxL Shell. However, the instructions in this chapter use the MaxL Shell.

This chapter contains the following sections:

Understanding Elements of a Query

In this section you will create a template to use as a basis for developing simple queries.

Most queries can be built upon the following grammatical framework:

```
SELECT
 {}
ON COLUMNS
FROM Sample.Basic
```

SELECT in line 1 is the keyword that begins the main body of all MDX statements.

The curly braces {} in line 2 are a placeholder for a *set*. In the above query, the set is empty, but the curly braces remain as a placeholder.

Exercise 1: Creating a Query Template

➤ To complete this exercise,

1. Create a folder to store sample queries which can be run against the Sample Basic database. For example, create a folder called "queries" under the `Essbase\Apps\Sample\Basic` directory of the Analytic Services installation.
2. Using a text editor, type the following code into a blank file:

   ```
   SELECT
    {}
   ON COLUMNS
   FROM Sample.Basic
   ```

3. Save the file as `qry_blank.txt` to your `queries` folder.

Note: If you are using the MDX Script Editor in Administration Services instead of a text editor, save the query as `qry_blank.MDX` from the editor instead.

Introduction to Sets and Tuples

A *set* is an ordered collection of one or more *tuples* that have the same dimensionality (see "Rules for Specifying Sets" on page 749 for an explanation of dimensionality).

A *tuple* is a way to refer to a member or a member combination from any number of dimensions. For example, in the Sample Basic database, Jan is a tuple, and so is (Jan, Sales), and so is ([Jan],[Sales],[Cola],[Utah],[Actual]).

The member name can be specified in the following ways:

1. By specifying the actual name or the alias; for example, `Cola`, `Actual`, `COGS`, and `[100]`.

   If the member name starts with number or contains spaces, it should be within braces; for example, `[100]`. Braces are recommended for all member names, for clarity and code readability.

   For attribute members, the long name (qualified to uniquely identify the member) should be used; for example, `[Ounces_12]` instead of just `[12]`.

2. By specifying dimension name or any one of the ancestor member names as a prefix to the member name; for example, `[Product].[100-10]` and `[Diet].[100-10]` This is a recommended practice for all member names, as it eliminates ambiguity and enables you to refer accurately to shared members.

   Note: Use no more than one ancestor in the member name qualification. Analytic Services returns an error if multiple ancestors are included. For example, `[Market].[New York]` is a valid name for New York, and so is `[East].[New York]`. However, `[Market].[East].[New York]` returns an error.

3. By specifying the name of a calculated member defined in the WITH section.

Recall that the curly braces {} in line 2 of your query template are a placeholder for a set. In this exercise, we will add a set to the query and run it.

Exercise 2: Running Your First Query

➤ To complete this exercise,

1. Open `qry_blank.txt`.
2. Recalling that a set can be as simple as one tuple, let us add `Jan` as a set to the query template. We must retain the curly braces, because these are required for all set specifications (except for sets that are produced by a function call).

   Type Jan inside the curly braces in line 2, as shown:

   ```
   SELECT
    {Jan}
   ON COLUMNS
   FROM Sample.Basic
   ```

3. Save the query as `ex2.txt`.
4. Make sure that Analytic Services is started (the `essbase.exe` process is running).
5. In order for Analytic Services to receive MDX statements, you must pass the statements to Analytic Services using either the MaxL Shell or MDX Script Editor in Administration Services. The examples in this chapter use the MaxL Shell.

   Start the MaxL Shell and log in with a valid user name and password. For example,

   ```
   essmsh -l admin passwd
   ```

6. Copy and paste the entire SELECT query into the MaxL Shell, but do not press the Enter key yet.
7. Type a semicolon at the end, anywhere after Basic but before pressing the Enter key. The semicolon is not part of MDX syntax requirements, but it is required by MaxL Shell to indicate the end of a statement that is ready to execute.

   Note: If you are using the MDX Script Editor in Administration Services, do not terminate with a semicolon.

8. Press the Enter key. You should see results similar to the following.

   Jan
   8024

Rules for Specifying Sets

As described in the previous section, a set is an ordered collection of one or more tuples.

For example, in the following query, {[100-10]} is a set consisting of one tuple.

```
SELECT
{[100-10]}
ON COLUMNS
FROM Sample.Basic
```

In the following query, {([100-10], [Actual])} is a also a set consisting of one tuple, though in this case, the tuple is not a single member name. Rather, ([100-10], [Actual]) represents a tuple consisting of members from two different dimensions, Product and Scenario.

```
SELECT
{([100-10], [Actual])}
ON COLUMNS
FROM Sample.Basic
```

When a set has more than one tuple, the following rule applies: In each tuple of the set, members must represent the same dimensions as do the members of other tuples of the set. Additionally, the dimensions must be represented in the same order. In other words, each tuple of the set must have the same *dimensionality.*

For example, the following set consists of two tuples of the same dimensionality.

```
{(West, Feb), (East, Mar)}
```

The following set breaks the dimensionality rule because Feb and Sales are from different dimensions.

```
{(West, Feb), (East, Sales)}
```

The following set breaks the dimensionality rule because although the two tuples contain the same dimensions, the order of dimensions is reversed in the second tuple.

```
{(West, Feb), (Mar, East)}
```

A set can also be a collection of sets, and it can also be empty (containing no tuples).

A set must be enclosed in curly brackets {} except in some cases where the set is represented by an MDX function which returns a set.

Introduction to Axis Specifications

An axis is a specification determining the layout of query results from a database. Axes fit into MDX queries as follows:

```
SELECT <axis> [, <axis>...]
FROM <database>
```

There must be at least one axis specified in any MDX query.

Up to 64 axes may be specified, beginning with AXIS(0) and continuing with AXIS(1)...AXIS(63). It is uncommon to use more than three axes. The order of axes is not important. However, when a set of axes 0 through *n* are specified, no axis between 0 and *n* should be skipped. Additionally, a dimension cannot appear on more than one axis.

The first five axes have keyword aliases:

`ON COLUMNS`	can be used in place of	`AXIS(0)`
`ON ROWS`	may replace	`AXIS(1)`
`ON PAGES`	may replace	`AXIS(2)`
`ON CHAPTERS`	may replace	`AXIS(3)`
`ON SECTIONS`	may replace	`AXIS(4)`

For example, in the query

```
SELECT {Jan} ON COLUMNS FROM Sample.Basic
```

the axis specification is `{Jan} ON COLUMNS`.

Exercise 3: Running A Two-Axis Query

➤ To complete this exercise,

1. Open `qry_blank.txt`.
2. Add a placeholder for a second axis, by adding the text shown in bold:

```
SELECT
 {}
ON COLUMNS,
 {}
ON ROWS
FROM Sample.Basic
```

Note: Remember to add the comma after ON COLUMNS.

3. Save the new query template as `qry_blank_2ax.txt`.
4. As the set specification for the column axis, enter the Product members 100-10 and 100-20. These member names contain special characters, so you will need to use braces. For example, add the text shown in bold:

```
SELECT
 {[100-10],[100-20]}
ON COLUMNS,
 {}
ON ROWS
FROM Sample.Basic
```

Note: In this chapter, the convention will be to enclose all member names in braces, even if they do not contain special characters. This convention is recommended.

5. As the set specification for the row axis, enter the Year members Qtr1 through Qtr4.

```
SELECT
 {[100-10],[100-20]}
ON COLUMNS,
 {[Qtr1],[Qtr2],[Qtr3],[Qtr4]}
ON ROWS
FROM Sample.Basic
```

6. Save the query as `ex3.txt`.

7. Paste the query into the MaxL Shell and run it, as described in "Exercise 2: Running Your First Query" on page 748.

   You should see results similar to the following.

	100-10	100-20
Qtr1	5096	1359
Qtr2	5892	1534
Qtr3	6583	1528
Qtr4	5206	1287

Exercise 4: Querying Multiple Dimensions on a Single Axis

➤ To complete this exercise,

1. Open `qry_blank_2ax.txt`.
2. On the column axis, specify two tuples, each of which is a member combination rather than a single member. You will need to enclose each tuple in parentheses, because there is more than one member represented in each tuple.

```
SELECT
 {([100-10],[East]), ([100-20],[East])}
ON COLUMNS,
 {}
ON ROWS
FROM Sample.Basic
```

3. On the row axis, specify four two-member tuples, nesting each Quarter with Profit:

```
SELECT
 {([100-10],[East]), ([100-20],[East])}
ON COLUMNS,
 {
  ([Qtr1],[Profit]), ([Qtr2],[Profit]),
  ([Qtr3],[Profit]), ([Qtr4],[Profit])
 }
ON ROWS
FROM Sample.Basic
```

4. Save the query as `ex4.txt`.
5. Paste the query into the MaxL Shell and run it, as described in "Exercise 2: Running Your First Query" on page 748.

   You should see results similar to the following.

		100-10	100-20
		East	**East**
Qtr1	**Profit**	2461	212
Qtr2	**Profit**	2490	303
Qtr3	**Profit**	3298	312
Qtr4	**Profit**	2430	287

Cube Specification

A cube specification is the part of the query that determines which database is being queried. The cube specification fits into an MDX query as follows:

```
SELECT <axis> [, <axis>...]
FROM <database>
```

The <database> section follows the FROM keyword and should consist of delimited or non delimited identifiers that specify an application name and a database name.

The first identifier should be an application name and the second one should be a database name. For example, all of the following are valid cube specifications:

- `FROM Sample.Basic`
- `FROM [Sample.Basic]`
- `FROM [Sample].[Basic]`
- `FROM'Sample'.'Basic'`

Using Functions to Build Sets

As an alternative to creating sets member-by-member or tuple-by-tuple, you can use a function that returns a set. MDX includes several functions that return sets, and also several functions that return other values. For a complete reference of MDX functions supported by Analytic Services, see the MaxL section of the online *Technical Reference*.

Exercise 5: Using the MemberRange Function

The MemberRange function returns a range of members inclusive of and between two specified members of the same generation. Its syntax is as follows:

`MemberRange (`*`member1, member2,`* `[,`*`layertype`*`])`

where the first argument you provide is the member that begins the range, and the second argument is the member that ends the range. The *layertype* argument is optional and will not be addressed here, for more information see the *Technical Reference*.

Note: An alternate syntax for MemberRange is to use a colon between the two members, instead of using the function name: *member1* : *member2.*

➤ To complete this exercise,

1. Open `qry_blank.txt`.
2. Delete the curly braces `{}`. Curly braces are not necessary when you are using a function to return the set.
3. Use the colon operator to select a member range of Qtr1 through Qtr4:

```
SELECT
 [Qtr1]:[Qtr4]
ON COLUMNS
FROM Sample.Basic
```

4. Paste the query into the MaxL Shell and run it, as described in "Exercise 2: Running Your First Query" on page 748.

   Qtr1, Qtr2, Qtr3, and Qtr4 are returned.

5. Use the MemberRange function to select the same member range, Qtr1 through Qtr4.

```
SELECT
 MemberRange([Qtr1],[Qtr4])
ON COLUMNS
FROM Sample.Basic
```

6. Run the query. The same results should be returned.
7. Save the query as `ex5.txt`.

Exercise 6: Using the CrossJoin Function

The CrossJoin function returns the cross product of two sets from different dimensions. Its syntax is as follows:

`CrossJoin(`*`set`*`,`*`set`*`)`

The CrossJoin function takes two sets from different dimensions as input and creates a set that is a cross product of the two input sets. This is useful for creating symmetric reports.

➤ To complete this exercise,

1. Open `qry_blank_2ax.txt`.
2. Delete the curly braces `{}` from the columns axis, and replace them with `CrossJoin()`.

```
SELECT
 CrossJoin ()
ON COLUMNS,
 {}
ON ROWS
FROM Sample.Basic
```

3. Add two comma-separated pairs of curly braces to use as placeholders for the two set arguments you will provide to the CrossJoin function:

```
SELECT
 CrossJoin ({}, {})
ON COLUMNS,
 {}
```

```
ON ROWS
FROM Sample.Basic
```

4. In the first set, specify the Product member `[100-10]`. In the second set, specify the Market members `[East]`, `[West]`, `[South]`, and `[Central]`.

```
SELECT
 CrossJoin ({[100-10]}, {[East],[West],[South],[Central]})
ON COLUMNS,
 {}
ON ROWS
FROM Sample.Basic
```

5. On the row axis, now use CrossJoin to cross a set of Measures members with a set containing Qtr1:

```
SELECT
 CrossJoin ({[100-10]}, {[East],[West],[South],[Central]})
ON COLUMNS,
  CrossJoin (
    {[Sales],[COGS],[Margin %],[Profit %]}, {[Qtr1]}
  )
ON ROWS
FROM Sample.Basic
```

6. Save the query as `ex6.txt`.
7. Paste the query into the MaxL Shell and run it, as described in "Exercise 2: Running Your First Query" on page 748.

   You should see results similar to the following.

		100-10 East	100-10 West	100-10 South	100-10 Central
Sales	**Qtr1**	5731	3493	2296	3425
COGS	**Qtr1**	1783	1428	1010	1460
Margin %	**Qtr1**	66.803	59.118	56.01	57.372
Profit %	**Qtr1**	45.82	29.974	32.448	24.613

When using CrossJoin, the order of arguments has an effect on the order of tuples in the output.

Exercise 7: Using the Children Function

The Children function returns a set of all child members of the given member. Its syntax is as follows:

```
Children (member)
```

Note: An alternate syntax for Children is to use it like an operator on the input member, as follows: *member.Children*. We will use the operator syntax in this exercise.

➤ To complete this exercise,

1. Open `ex6.txt`. You will use the Children function to introduce a shortcut in the first axis specification.
2. In the second set of the column axis specification, replace `[East],[West],[South],[Central]` with `[Market].Children`.

```
SELECT
 CrossJoin ({[100-10]}, {[Market].Children})
ON COLUMNS,
  CrossJoin (
    {[Sales],[COGS],[Margin %],[Profit %]}, {[Qtr1]}
  )
ON ROWS
FROM Sample.Basic
```

3. Save the query as `ex7.txt`.
4. Paste the query into the MaxL Shell and run it, as described in "Exercise 2: Running Your First Query" on page 748.
5. You should see the same results as were returned for "Exercise 6: Using the CrossJoin Function" on page 755.

Working with Levels and Generations

In MDX, the term *layer* is used to refer to generations and levels in an Analytic Services hierarchy.

In Analytic Services, generation numbers begin counting with 1 at the dimension name; higher generation numbers are those that are closest to leaf members in a hierarchy.

Level numbers begin with 0 at the leaf-most part of the hierarchy, and the highest level number is a dimension name.

A number of MDX functions take a layer you specify as an input argument, and perform set operations based on the generation or level represented in the layer argument.

You can specify a layer argument in the following ways:

- By generation or level name; for example, `States` or `Regions`.
- The dimension name along with the generation or level name; for example, `Market.Regions` and `[Market].[States]`.
- The Levels function with a dimension and a level number as input. For example, `[Year].Levels(0)`.
- The Level function with a member as input. For example, `[Qtr1].Level` returns the level of quarters in Sample Basic, which is level 1 of the Market dimension.
- The Generations function with a dimension and a generation number as input. For example, `[Year].Generations (3)`.
- The Generation function with a member as input. For example, `[Qtr1].Generation` returns the generation of quarters in Sample Basic, which is generation 2 of the Market dimension.

Note: In the Sample Basic database, Qtr1 and Qtr4 are in the same layer.This means that Qtr1 and Qtr4 are also in the same generation. However, in a different database with a ragged hierarchy, Qtr1 and Qtr4 might not necessarily be in the same level even though they are in the same generation. For example, if the hierarchy of Qtr1 drills down to weeks and the hierarchy of Qtr4 stops at months, then Qtr1 is one level higher than Qtr4, but they are still in the same layer.

Exercise 8: Using the Members Function

The Members function can be used to return all members of a specified generation or level. Its syntax when used with a layer argument is as follows:

`Members (`*`layer`*`)`

where the *layer* argument you provide indicates the generation or level of members you want returned.

Note: An alternate syntax for Members is *layer*.Members.

➤ To complete this exercise,

1. Open `qry_blank.txt`.
2. Delete the curly braces `{}`.
3. Use the Members function and the Levels function to select all level-0 members in the Market dimension of Sample Basic:

```
SELECT
  Members(Market.levels(0))
ON COLUMNS
FROM Sample.Basic
```

4. Paste the query into the MaxL Shell and run it, as described in "Exercise 2: Running Your First Query" on page 748.

   All of the states in the Market dimension are returned.

5. Save the query as `ex8.txt`.

Using a Slicer Axis to Set Query Point-of-View

A *slicer* axis is a way of limiting a query to apply only to a specific area of the database. The optional slicer, if used, must be in the WHERE section of an MDX query. Furthermore, the WHERE section must be the last component of the query, following the cube specification (the FROM section):

```
SELECT {set}
ON axes
FROM database
WHERE slicer
```

The slicer axis is used to set the context of the query, and is usually the default context for all the other axes.

For example, if you want a query to select only Actual Sales in the Sample Basic database, excluding budgeted sales, the WHERE clause might look like the following:

```
WHERE ([Actual], [Sales])
```

Because (Actual, Sales) is specified in the slicer axis, it is not necessary to include them in the ON AXIS(n) set specifications.

Exercise 9: Limiting the Results with a Slicer Axis

➤ To complete this exercise,

1. Open `ex6.txt`.
2. Paste the query into the MaxL Shell and run it, as described in "Exercise 2: Running Your First Query" on page 748.

   Note the results in one of the data cells; for example, notice that the first tuple, `([100-10],[East],[Sales],[Qtr1])`, has a value of 5731.
3. Add a slicer axis to limit the data returned to only budgeted values.

```
SELECT
 CrossJoin ({[100-10]}, {[East],[West],[South],[Central]})
ON COLUMNS,
  CrossJoin (
    {[Sales],[COGS],[Margin %],[Profit %]}, {[Qtr1]}
  )
ON ROWS
FROM Sample.Basic
WHERE (Budget)
```

4. Paste the query into the MaxL Shell and run it.

   Note that the results are different.
5. Save the query as `ex9.txt`.

Common Relationship Functions

The following relationship functions return **sets** based on member relationships in the database outline:

Relationship Function	Description
Children	Returns the children of the input member.
Siblings	Returns the siblings of the input member.
Descendants	Returns the descendants of a member, with varying options.

The following functions are also relationship functions, but they return a single member rather than a set:

Relationship Function	Description
Ancestor	Returns an ancestor at the specified layer.
Cousin	Returns a child member at the same position as a member from another ancestor.
Parent	Returns the parent of the input member.
FirstChild	Returns the first child of the input member.
LastChild	Returns the last child of the input member.
FirstSibling	Returns the first child of the input member's parent.
LastSibling	Returns the last child of the input member's parent.

For examples using relationship functions, see the MDX examples in the MaxL section of the *Technical Reference*.

Performing Set Operations

The following set functions are characterized by the fact that they operate on input sets without deriving further information from a database:

Pure Set Function	Description
CrossJoin	Returns a cross-section of two sets from different dimensions.
Distinct	Deletes duplicate tuples from a set.
Except	Returns a subset containing the differences between two sets.
Generate	An iterative function. For each tuple in *set1*, returns *set2*.
Head	Returns the first *n* members or tuples present in a set.
Intersect	Returns the intersection of two input sets.

Pure Set Function	Description
Subset	Returns a subset from a set, in which the subset is a numerically specified range of tuples.
Tail	Returns the last *n* members or tuples present in a set.
Union	Returns the union of two input sets.

Exercise 10: Using the Intersect Function

The Intersect function returns the intersection two input sets, optionally retaining duplicates. Its syntax is as follows:

```
Intersect (set, set [,ALL])
```

You can use the Intersect function to compare sets by finding tuples that are present in both sets.

➤ To complete this exercise,

1. Open `qry_blank.txt`.
2. Delete the curly braces `{}` from the axis, and replace them with `Intersect ()`.

```
SELECT
 Intersect (

 )
ON COLUMNS
FROM Sample.Basic
```

3. Add two comma-separated pairs of curly braces to use as placeholders for the two set arguments you will provide to the Intersect function:

```
SELECT
 Intersect (
 { },
 { }
 )
ON COLUMNS
FROM Sample.Basic
```

4. Specify children of East as the first set argument.

```
SELECT
 Intersect (
 { [East].children },
 { }
 )
ON COLUMNS
FROM Sample.Basic
```

5. For the second set argument, specify all members of the Market dimension that have a UDA of "Major Market."

   Note: To learn more about how the UDA function works, see the *Technical Reference*.

```
SELECT
 Intersect (
 { [East].children },
 { UDA([Market], "Major Market") }
 )
ON COLUMNS
FROM Sample.Basic
```

6. Paste the query into the MaxL Shell and run it, as described in "Exercise 2: Running Your First Query" on page 748.

   The results will be all children of East that have a UDA of "Major Market":

New York	Massachusetts	Florida
8202	6712	5029

7. Save the query as `ex10.txt`.

Exercise 11: Using the Union Function

The Union function joins two input sets, optionally retaining duplicates. Its syntax is as follows:

```
Union (set, set [,ALL])
```

You can use the Union function to lump two sets together into one set.

➤ To complete this exercise,

1. Open `ex10.txt`.
2. Replace Intersect with Union, leaving everything else the same.
3. Paste the query into the MaxL Shell and run it.

   If you compare the results with the results of the Intersect function in the previous exercise, you will see that while Intersect returns a set containing only those children of East that have a UDA of “Major Market,” Union returns {all children of east) + (all Market Members that have a UDA of “Major Market.”)
4. Save the query as `ex11.txt`.

For more examples using pure set-operative functions, see the *Technical Reference*.

Creating and Using Named Sets and Calculated Members

Calculated members and named sets are logical entities in query which can be used multiple times during the life of the query. Calculated members and named sets can save time in lines of code written as well as in execution time.The optional WITH section at the beginning of an MDX query is where you define the calculated members and/or named sets.

Calculated Members

A calculated member is a hypothetical member existing for the duration of the query execution. Calculated members enable you to perform complex analysis without the necessity of adding new members to the database outline. Essentially, calculated members are a storage place for calculation results performed on real members.

You can give a calculated member any name you want, with the following guidelines:

- You must associate the calculated member with a dimension; for example, to associated the member MyCalc with the Measures dimension, you would name it `[Measures].[MyCalc]`.

- Do not use real member names to name calculated members; for example, do not name a calculated member `[Measures].[Sales]`, because Sales already exists in the Measures dimension.

Whenever you use multiple calculated members to create ratios or custom totals, it is good practice to set the solve order for each calculated member. For more information about solve order, see Grammar Rules > With Specification in the MDX section of the *Technical Reference*.

Exercise 12: Creating a Calculated Member

This exercise will include the Max function, a common function for calculations. The Max function returns the maximum of values found in the tuples of a set. Its syntax is as follows:

`Max (set, numeric_value)`

➤ To complete this exercise,

1. Open `qry_blank_2ax.txt`.
2. On the row axis set, specify the children of Product.

```
SELECT
 {}
ON COLUMNS,
 {[Product].children}
ON ROWS
FROM Sample.Basic
```

3. At the beginning of the query, add a placeholder for the calculated member specification:

```
WITH MEMBER [].[]
 AS ''
SELECT
 {}
ON COLUMNS,
 {[Product].children}
ON ROWS
FROM Sample.Basic
```

4. You will associate the calculated member with the Measures dimension, and name it Max Qtr2 Sales. To do this, add this information to the calculated member specification:

```
WITH MEMBER [Measures].[Max Qtr2 Sales]
 AS ''
SELECT
 {}
ON COLUMNS,
 {[Product].children}
ON ROWS
FROM Sample.Basic
```

5. After the AS keyword and inside the single quotation marks, you will define the logic for the calculated member named Max Qtr2 Sales. Use the Max function with the first argument being the set to evaluate (Qtr2), and the second argument being the measure to evaluate (Sales).

```
WITH MEMBER [Measures].[Max Qtr2 Sales]
 AS '
  Max (
    {[Year].[Qtr2]},
    [Measures].[Sales]
  )'
SELECT
 {}
ON COLUMNS,
 {[Product].children}
ON ROWS
FROM Sample.Basic
```

6. Now that the calculated member Max Qtr2 Sales is defined in the WITH section, in order to use it in a query, you must include it on one of the axes in the SELECT portion of the query. Select the pre-defined calculated member on the columns axis:

```
WITH MEMBER [Measures].[Max Qtr2 Sales]
 AS '
  Max (
    {[Year].[Qtr2]},
    [Measures].[Sales]
  )'
SELECT
 {[Measures].[Max Qtr2 Sales]}
ON COLUMNS,
```

```
 {[Product].children}
ON ROWS
FROM Sample.Basic
```

7. Paste the query into the MaxL Shell and run it, as described in "Exercise 2: Running Your First Query" on page 748.

   You should see results similar to the following.

	Max Qtr2 Sales
100	27187
200	27401
300	25736
400	21355
Diet	26787

8. Save the query as `ex12.txt`.

Named Sets

A named set is a set specification just like those you would define in the SELECT axes, except you define the sets in the WITH section of the query, and associate them with a name. This is useful because you can reference the set by name when building the SELECT section of the query.

For example, a named set called `Best5Prods` identifies a set of the five top-selling products in December:

```
WITH
SET [Best5Prods] AS
 'Topcount (
   [Product].members,
   5,
   ([Measures].[Sales], [Scenario].[Actual],
    [Year].[Dec])
  )'
SELECT [Best5Prods] ON AXIS(0),
{[Year].[Dec]} ON AXIS(1)
FROM Sample.Basic
```

Using Iterative Functions

The following functions loop through sets of data and perform search conditions that you specify, with results that you specify.

Function	Description
Filter	Returns the subset of tuples in *set* for which the value of the search condition is TRUE.
IIF	Performs a conditional test, and returns an appropriate numeric expression or set depending on whether the test evaluates to TRUE or FALSE.
Case	Performs conditional tests and returns the results you specify.
Generate	An iterative function. For each tuple in *set1*, returns *set2*.

Filter Function Example

The following query returns all Market dimension members for which the expression `IsChild([Market].CurrentMember,[East])` returns TRUE; in other words, the query returns all children of East.

```
SELECT
 Filter([Market].Members,
        IsChild([Market].CurrentMember,[East])
 )
ON COLUMNS
FROM Sample.Basic
```

The Filter function in MDX is comparable to the RESTRICT command in Report Writer.

For more examples of Filter and other iterative functions, see the *Technical Reference*.

Working With Missing Data

When you are querying on a database, you can use the NON EMPTY keywords at the beginning of an axis specification to prevent cells containing no value from being included the result of the query.

The axis specification syntax including NON EMPTY is shown next:

```
<axis_specification> ::=
       [NON EMPTY] <set> ON
       COLUMNS | ROWS | PAGES | CHAPTERS |
      SECTIONS | AXIS (<unsigned_integer>)
```

Including the optional keywords NON EMPTY before the set specification in an axis causes suppression of slices in that axis that would contain entirely #MISSING values.

For any given tuple on an axis (such as `(Qtr1, Actual)`), a *slice* consists of the cells arising from combining this tuple with all tuples of all other axes. If all of these cell values are #MISSING, the NON EMPTY keyword causes the tuple to be eliminated.

For example, if even one value in a row is not empty, the entire row is returned. Including NON EMPTY at the beginning of the row axis specification would eliminate the following row slice from the set returned by a query:

Qtr1					
Actual	#Missing	#Missing	#Missing	#Missing	#Missing

In addition to suppressing missing values with NON EMPTY, you can use the following MDX functions to handle #MISSING results:

- CoalesceEmpty, which searches numeric value expressions for non #MISSING values
- IsEmpty, which returns TRUE if the value of an input numeric-value-expression evaluates to #MISSING.
- Avg, which omits missing values from averages unless you use the optional IncludeEmpty flag.

For more information, see the MDX section of the *Technical Reference*.

Querying for Properties

In MDX, *properties* describe certain characteristics of data and metadata. MDX enables you to write queries that use properties to retrieve and analyze data. Properties can be intrinsic or custom.

Intrinsic properties are defined for members in all dimensions. The intrinsic member properties defined for all members in an Analytic Services database outline are MEMBER_NAME, MEMBER_ALIAS, LEVEL_NUMBER, and GEN_NUMBER.

MDX in Analytic Services supports two types of custom properties: attribute properties and UDA properties. Attribute properties are defined by the attribute dimensions in an outline. In the Sample Basic database, the Pkg Type attribute dimension describes the packaging characteristics of members in the Product dimension. This information can be queried in MDX using the property name [Pkg Type].

Attribute properties are defined only for specific dimensions and only for a specific level in each dimension. For example, in the Sample Basic outline, `[Ounces]` is an attribute property defined only for members in the Product dimension, and this property has valid values only for the level-0 members of the Product dimension. The [Ounces] property does not exist for other dimensions, such as Market. The [Ounces] property for a non level-0 member in the Product dimension is a NULL value. The attribute properties in an outline are identified by the names of attribute dimensions in that outline.

The custom properties also include UDAs. For example, [Major Market] is a UDA property defined on Market dimension members. It returns a TRUE value if [Major Market] UDA is defined for a member, and FALSE otherwise.

Querying for Member Properties

Properties can be used inside an MDX query in two ways. In the first approach, you can list the dimension and property combinations for each axis set. When a query is executed, the specified property is evaluated for all members from the specified dimension and included in the result set.

For example, on the column axis, the following query will return the GEN_NUMBER information for every Market dimension member. On the row axis, the query returns MEMBER_ALIAS information for every Product dimension member.

```
SELECT
 [Market].Members
    DIMENSION PROPERTIES [Market].[GEN_NUMBER] on columns,
 Filter ([Product].Members, Sales > 5000)
    DIMENSION PROPERTIES [Product].[MEMBER_ALIAS] on rows
from Sample.Basic
```

When querying for member properties using the DIMENSION PROPERTIES section of an axis, a property can be identified by the dimension name and the name of the property, or just by using the property name itself. When a property name is used by itself, that property information is returned for all members from all dimensions on that axis, for which that property applies. In the following query. the MEMBER_ALIAS property is evaluated on the row axis for both Year and Product dimensions.

```
SELECT
 [Market].Members
    DIMENSION PROPERTIES [Market].[GEN_NUMBER] on columns,
 CrossJoin([Product].Children, Year.Children)
    DIMENSION PROPERTIES [MEMBER_ALIAS] on rows
from Sample.Basic
```


In the second approach, properties can be used inside value expressions in an MDX query. For example, you can filter a set based on a value expression that uses properties of members in the input set.

The following query returns all caffeinated products that are packaged in cans.

```
SELECT
    Filter([Product].levels(0).members,
              [Product].CurrentMember.Caffeinated and
              [Product].CurrentMember.[Pkg Type] = "Can")
          Dimension Properties
          [Caffeinated], [Pkg Type] on columns
 FROM Sample.Basic
```

The following query calculates the value [BudgetedExpenses] based on whether the current Market is a major market, using the UDA [Major Market].

```
WITH
 MEMBER [Measures].[BudgetedExpenses] AS
 'IIF([Market].CurrentMember.[Major Market],
  [Marketing] * 1.2, [Marketing])'
SELECT
 {[Measures].[BudgetedExpenses]} ON COLUMNS,
```

```
  [Market].Members ON ROWS
FROM Sample.Basic
WHERE ([Budget])
```

The Value Type of Properties

The value of an MDX property in Analytic Services can be a numeric, Boolean, or string type. MEMBER_NAME and MEMBER_ALIAS properties return string values. LEVEL_NUMBER and GEN_NUMBER properties return numeric values.

The attribute properties return numeric, Boolean, or string values based on the attribute dimension type. For example, in Sample Basic, the [Ounces] attribute property is a numeric property. The [Pkg Type] attribute property is a string property. The [Caffeinated] attribute property is a Boolean property.

Analytic Services allows attribute dimensions with date types. The date type properties are treated as numeric properties in MDX. When comparing these property values with dates, you need to use the TODATE function to convert date strings to numeric before comparison.

The following query returns all Product dimension members that have been introduced on date 03/25/1996. Since the property [Intro Date] is a date type, the TODATE function must be used to convert the date string “03-25-1996” to a number before comparing it.

```
SELECT
 Filter ([Product].Members,
         [Product].CurrentMember.[Intro Date] =
         TODATE("mm-dd-yyyy","03-25-1996"))ON COLUMNS
FROM Sample.Basic
```

When a property is used in a value expression, you must use it appropriately based on its value type: string, numeric, or Boolean.

You can also query attribute dimensions with numeric ranges.

The following query retrieves Sales data for Small, Medium and Large population ranges.

```
SELECT
 {Sales} ON COLUMNS,
 {Small, Medium, Large} ON ROWS
FROM Sample.Basic
```

When attributes are used as properties in a value expression, you can use range members to check whether a member's property value falls within a given range, using the IN operator.

For example, the following query returns all Market dimension members with the population range in Medium:

```
SELECT
  Filter(
       Market.Members, Market.CurrentMember.Population
       IN "Medium"
  )
ON AXIS(0)
FROM Sample.Basic
```

NULL Property Values

Not all members may have valid values for a given property name. For example, the MEMBER_ALIAS property returns an alternate name for a given member as defined in the outline; however, not all members may have aliases defined. In these cases A NULL value is be returned for those members that do not have aliases.

In the following query,

```
SELECT
  [Year].Members
   DIMENSION PROPERTIES [MEMBER_ALIAS]
ON COLUMNS
FROM Sample.Basic
```

none of the members in the Year dimension have aliases defined for them. Therefore, the query returns NULL values for the MEMBER_ALIAS property for members in the Year dimension.

The attribute properties are defined for members of a specific dimension and a specific level in that dimension. In the Sample Basic database, the [Ounces] property is defined only for level-0 members of the Product dimension.

Therefore, if you query for the [Ounces] property of a member from the Market dimension, as shown in the following query, you will get a syntax error:

```
SELECT
    Filter([Market].members,
           [Market].CurrentMember.[Ounces] = 32) ON COLUMNS
FROM Sample.Basic
```

Additionally, if you query for the [Ounces] property of a non level-0 member of the dimension, you will get a NULL value.

When using property values in value expressions, you can use the function IsValid() to check for NULL values. The following query returns all Product dimension members with an [Ounces] property value of 12, after eliminating members with NULL values.

```
SELECT
  Filter([Product].Members,
         IsValid([Product].CurrentMember.[Ounces]) AND
         [Product].CurrentMember.[Ounces] = 12)
ON COLUMNS
FROM Sample.Basic
```

Index

Symbols

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Numerics

A

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

B

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

C

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Z

Essbase Analytic Services

Release 7.1

Database Administrator's Guide

Volume III: Optimization and System Administration

Hyperion Solutions Corporation

Hyperion Solutions Corporation
1344 Crossman Avenue
Sunnyvale, California 94089

Printed in the U.S.A.

Contents

Part VI

Designing and Managing a Security System

This part describes how to control access to data in Analytic Services applications by designing and building a security system:

- Chapter 36, "Managing Security for Users and Applications," describes how to create a security plan to manage security at the user and group layer, the global access layer, and the server level.
- Chapter 37, "Controlling Access to Database Cells," describes how to define security filters and assign them to users.
- Chapter 38, "Security Examples," contains detailed examples that illustrate how to implement a security system for Analytic Services applications.

Chapter

36

Managing Security for Users and Applications

Analytic Services provides a comprehensive system for managing access to applications, databases, and other objects. The Analytic Services security system provides protection in addition to the security available through your local area network.

This chapter contains the following sections:

Understanding Security and Permissions

The Analytic Services security system addresses a wide variety of database security needs with a multi layered approach to enable you to develop the best plan for your environment. Various levels of permission can be granted to users and groups or defined at the system, application, or database scope. You can apply security in the following ways:

- Users and groups.

  To grant permissions to individual users and groups of users. When higher, these permissions take precedence over minimum permissions defined for applications and databases. Ordinary users have no inherent permissions. Permissions can be granted to users and groups by editing the users and groups or by using the **grant** statement in MaxL DDL (data-definition language). For an explanation of the methods used to grant permissions, see "Granting Permissions to Users and Groups" on page 840.

  You can create users who log on using the parameters of an external authentication repository instead of the Analytic Services password. If you want users to use an outside authentication repository such as LDAP, you must implement the Hyperion security platform and create the Analytic Services users with a reference to the security platform. For information about the security platform, see "Using the Hyperion Security Platform for External Authentication" on page 848.

- Application and database settings.

  To set common permissions for all users of an application or database, you can set minimum permissions that all users can have at each application or database scope. Users and groups with lower permissions than the minimum gain access; users and groups with higher granted permissions are not affected. You can also temporarily disable different kinds of access using application settings. For a detailed explanation of the use of application and database settings, see "Managing Global Security for Applications and Databases" on page 849.

- Server-wide settings.

  Create and manage login restrictions for the entire Analytic Server. View and terminate current sessions and requests running on the entire Analytic Server or only on particular applications and databases. For a comprehensive discussion of how to manage the activities of users, see "Managing User Activity on the Analytic Server" on page 856.

- Database filters.

  Define database permissions that users and groups can have for particular members, down to the individual data value (cell). To learn more about how and why to use filters, see Chapter 37, "Controlling Access to Database Cells."

Table 28 describes all security permissions and the tasks that can be performed with those permissions.

Table 28: Description of Analytic Services Permissions

Permission	Affected Scope	Description
No Access or None	Entire system, application, or database	No inherent access to any users, groups, or data values, unless access is granted globally or by means of a filter. No Access is the default when creating an ordinary user. Users with No Access permissions can change their own passwords.
Read	Database	Ability to read data values.
Write	Database	Ability to read and update data values.
MetaRead	Database	Ability to read metadata (dimension and member names) and update data for the corresponding member specification.
Execute (or Calculate)	Entire system, application, database, or single calculation	Ability to calculate, read, and update data values for the assigned scope, using the assigned calculation. Supervisors, application designers for the application, and database designers for the database can run calculations without being granted execute access.
Database Designer	Database	Ability to modify outlines, create and assign filters, alter database settings, and remove locks/terminate sessions and requests on the database. A user with Database Designer permission in one database does not necessarily have that permission in another.

Table 28: Description of Analytic Services Permissions (Continued)

Permission	Affected Scope	Description
Application Designer	Application	Ability to create, delete, and modify databases within the assigned application. Ability to modify the application settings, including minimum permissions, remove locks on the application, terminate sessions and requests on the application, and modify any object within the application. You cannot create or delete an application unless you also have been granted the system-level Create/Delete Applications permission. A user with Application Designer permission in one application does not necessarily have that permission in another.
Filter Access	Database	Ability to access specific data and metadata according to the restrictions of a filter assigned to the user or group. The filter definition specifies, for subsets of a database, whether Read, Write, No Access, or MetaRead is allowed for each subset. A user or group can be granted only one filter per database. Filters can be used in conjunction with other permissions. For a comprehensive discussion of filters, see Chapter 37, "Controlling Access to Database Cells."
Create/delete applications	Entire system	Ability to create and delete applications and databases within those applications, and control permissions, locks, and resources for applications created. Includes designer permissions for the applications and databases created by this user.
Create/Delete Users, Groups	Entire system	Ability to create, delete, edit, or rename all users and groups having equal or lesser permissions than their own.
Supervisor	Entire system	Full access to the entire system and all users and groups.

Creating Users and Groups

When you create a user or a group in Analytic Services, you define a security profile. The security profile is where you define the extent of the permissions that users and groups have in dealing with each other and in accessing applications and databases. This section contains the following sections:

If you are using Administration Services, you also need to create users on the Administration Server. For more information, see "About Administration Services Users" on page 119.

Creating Users

To create a user means to define the user name, password, and permission. You can also specify group membership for the user, and you can specify that the user is required to change the password at the next login attempt, or that the user name is disabled, preventing the user from logging on.

User names can contain only characters defined within the code page referenced by the ESSLANG variable and they cannot contain the backslash (\) character. User names must begin with a letter or a number.

➤ To create a new user, use either of the following methods:

Tool	Topic	Location
Administration Services	Creating Users on Analytic Servers	*Essbase Administration Services Online Help*
MaxL	**create user**	*Technical Reference*

For example, to create a user named admin and grant that user Supervisor permissions, use the following MaxL statements:

```
create user admin identified by 'password';
grant supervisor to admin;
```

Creating Groups

A group is a collection of users who share the same minimum access permissions. Placing users in groups can save you the time of assigning identical permissions to users again and again.

Note: A member of a group may have permissions beyond those assigned to the group, if permissions are also assigned individually to that user.

The process for creating, editing, or copying groups is the same as that for users, except that there are no group passwords. You define group names and permissions just as you do for users.

Note: A group name may not contain a backslash (\).

When you create a new user, you can assign the user to a group. Similarly, when you create a new group, you can assign users to the group. You must define a password for each user; there are no passwords for groups.

➤ To create groups, use either of the following methods:

Tool	Topic	Location
Administration Services	Creating Groups on Analytic Servers	*Essbase Administration Services Online Help*
MaxL	**create group**	*Technical Reference*

Granting Permissions to Users and Groups

You can define security permissions for individual users and groups. Groups are collections of users that share the same minimum permissions. Users inherit all permissions of the group and can additionally have access to permissions that exceed those of the group.

Permissions can be granted to users and groups in the following ways:

- Specifying a user or group type upon creation or by editing the user or group. This method specifies system-level permissions that span all applications and databases. For descriptions of the types of users and groups, see "Assigning User and Group Types" on page 841.

- Granting permission to access applications or databases. For an explanation of how to grant resource-specific permissions, see "Granting Application and Database Access to Users and Groups" on page 842.
- Granting designer permissions to users or groups. For an explanation of situations requiring the granting of designer permissions, see "Granting Designer Permissions to Users and Groups" on page 844.

Assigning User and Group Types

One way to assign permissions to users and groups is to define user and group types when you create or edit (modify the permissions of) the users and groups.

In Administration Services, users and groups can be created in different ways to specify their system-level permissions. These methods are represented in Administration Services Console as user types. In MaxL, user types do not exist; instead you grant the permissions after the user is created.

In Administration Services, users can be created with the following types:

- Supervisor.

  A user or group with Supervisor permission has full access to the entire system and all users and groups. The user who installs Analytic Services on the server is designated the System Supervisor for that server. Analytic Services requires that at least one user on each server has Supervisor permission. Therefore, you cannot delete or downgrade the permission of the last supervisor on the server.

- User.

  Users or groups with ordinary permission have no inherent access to any users, groups, or resources. This type of user is the default user.

- Users with Create/Delete Users, Groups permission.

  This type of user or group can create, delete, edit, or rename users and groups with equal or lower permissions only.

- Users with Create/Delete Applications permission.

  This type of user or group can create and delete applications and control permissions and resources applicable to those applications or databases they created.

Users with Create/Delete Applications permission cannot create or delete users, but they can manage application-level permission for those applications that they have created. For a comprehensive discussion of application-level permission, see "Managing Global Security for Applications and Databases" on page 849.

For instructions about creating users and groups, see "Creating Users" on page 839 and "Creating Groups" on page 840.

Granting Application and Database Access to Users and Groups

If you need to grant resource-specific permissions to users and groups that are not implied in any user types, you can grant the specific application or database permissions to users when creating or editing them in Administration Services. Using MaxL, you grant the permissions after the user is created by using the **grant** statement.

You can grant or modify user and group application and database permissions from an edit-user standpoint or from an application or database security perspective. The results are the same.

Note: If a user has insufficient permission to access the data in a database, the value does not show up in queries, or shows up as #NOACCESS.

There is no need to grant permissions to users or groups that are already Supervisors—they have full permissions to all resources on the Analytic Server. For a given database, users or groups can also be granted any of the following permissions:

Table 29: Permissions Available at the Database Scope

Database permission	Description
None	Indicates no access to any object or data value in a database.
Filter Access	Indicates that data and metadata access is restricted to those filters assigned to the user. (For a comprehensive discussion of filters, see Chapter 37, "Controlling Access to Database Cells.") The Filter check box grants a filter object to a user or group. A user or group can be granted only one filter per database. Selecting this option or any other option except None enables the selection of a filter object from the list box.
Read Only	Indicates read permission, that is, the ability to retrieve all data values. Report scripts can also be run.
Read / Write	Indicates that all data values can be retrieved and updated (but not calculated). The user can run, but cannot modify, Analytic Services objects.
MetaRead	Indicates that metadata (dimension and member names) can be retrieved and updated for the corresponding member specification.
Calculate	Indicates that all data values can be retrieved, updated, and calculated with the default calculation or any calculation for which the user has been granted permission to execute.
Database Designer	Indicates that all data values can be retrieved, updated, and calculated. In addition, all database-related files can be modified.

➤ To grant or modify application or database permissions for a user or group, use either of the following methods:

Tool	Topic	Location
Administration Services	Managing User/Group Permissions for Applications and Databases	*Essbase Administration Services Online Help*
MaxL	To grant permissions: **grant** To change the user type or group: **alter user**	*Technical Reference*

Granting Designer Permissions to Users and Groups

Users and groups can be granted Application Designer or Database Designer permission for particular applications or databases. These permissions are useful for assigning administrative permissions to users who need to be in charge of particular applications or databases, but who only need ordinary user permissions for other purposes.

You need to grant database access to other users if any of the following conditions apply:

- The users have not been granted sufficient user permission to access databases.
- The database in question does not allow users sufficient access through its minimum-permission settings.
- The users do not have sufficient access granted to them through filters.

For references to methods you can use to grant Designer permissions to a user or group, see "Granting Application and Database Access to Users and Groups" on page 842.

Managing Users and Groups

To help manage security between users and groups, the following user-management tasks are available at varying degrees to users with different permissions:

Viewing Users and Groups

➤ To view users and groups, use either of the following methods:

Tool	Topic	Location
Administration Services	Enterprise View > Security node > Users node or Groups node	*Essbase Administration Services Online Help*
MaxL	**display user** or **display group**	*Technical Reference*

Editing Users

To edit a user means to modify the security profile established when the user was created. For information about changing user passwords, see "Propagating Password Changes" on page 859.

➤ To change a password or other user properties, use either of the following methods:

Tool	Topic	Location
Administration Services	Editing User Properties	*Essbase Administration Services Online Help*
MaxL	**alter user**	*Technical Reference*

Editing Groups

To edit a group means to modify the security profile established when the group was created.

➤ To view or change group membership, use either of the following methods:

Tool	Topic	Location
Administration Services	Editing Group Properties	*Essbase Administration Services Online Help*
MaxL	**display user in group** or **alter group**	*Technical Reference*

Copying an Existing Security Profile

An easy way to create a new user with the same permissions as another user is to copy the security profile of an existing user. The new user is assigned the same user type, group membership, and application/database access as the original user.

You can also create new groups by copying the security profile of an existing group. The new group is assigned the same group type, user membership, and application access as the original group.

You can copy users and groups on the same Analytic Server or from one Analytic Server to another, according to your permissions. You can also migrate users and groups across servers along with an application. For more information, see "Copying Users" in *Essbase Administration Services Online Help*.

To copy a user or group means to duplicate the security profile of an existing user or group, and to give it a new name. It is helpful to copy users and groups because it saves you the time of reassigning permissions in cases where you want them to be identical.

Note: Copying removes any security permissions the creator does not have from the copy. For example, a user with Create/Delete Users permission cannot create a new supervisor by copying the profile of an existing supervisor.

➤ To create a new user or group by copying the security profile of an existing user or group, use either of the following methods:

Tool	Topic	Location
Administration Services	Copying Users and Copying Groups	*Essbase Administration Services Online Help*
MaxL	**create user** or **create group**	*Technical Reference*

Deleting Users and Groups

➤ To delete users and groups, use either of the following methods:

Tool	Topic	Location
Administration Services	Deleting Users and Deleting Groups	*Essbase Administration Services Online Help*
MaxL	**drop user** or **drop group**	*Technical Reference*

Renaming Users and Groups

➤ To rename users and groups, use either of the following methods:

Note: A group name may not contain a backslash (\).

Tool	Topic	Location
Administration Services	Renaming Users and Renaming Groups	*Essbase Administration Services Online Help*
MaxL	**alter user** and **alter group**	*Technical Reference*

Using the Hyperion Security Platform for External Authentication

External authentication means that the user login information needed by Analytic Services is maintained in a central authentication directory, such as Lightweight Directory Access Protocol (LDAP) Directory, Microsoft Active Directory, or Windows NT LAN Manager.

An authentication directory is a centralized store of user information such as login names and passwords, and other corporate information. The repository functions like a telephone directory. The authentication directory probably contains much more than user names and passwords; for example, it may include e-mail addresses, employee IDs, job titles, access rights, and telephone numbers. It may also contain objects other than users; for example, it may contain information about corporate locations or other entities.

To use the security platform for Analytic Services, your organization must already have an authentication directory that contains centralized user information. Additionally, you must employ an XML-based security configuration file to specify correct information pertaining to your corporate authentication directory.

The following types of authentication repositories are supported by the security platform:

- Windows NT LAN Manager (NTLM)
- Lightweight Directory Access Protocol (LDAP)
- Microsoft Active Directory server (MSAD)

➤ To learn more about implementing the Hyperion security platform,

1. Check the *Essbase Analytic Services Installation Guide* to make sure your required platform and authentication directory are supported.
2. See the help topic "Configuration for External Authentication," which is in the "Security Platform Reference" section of the *Technical Reference*. Follow the configuration guidelines in that document.
3. To manage external authentication of users using Administration Services, see also "Managing External Authentication" in the *Essbase Administration Services Online Help*.

Note: Prior to this release, Analytic Services provided support for external authentication using built-in modules supplied as parameters to the AUTHENTICATIONMODULE `essbase.cfg` setting. Those modules did not include the ability to share the same external authentication scheme with other Hyperion software applications or to enable single sign-on to Analytic Services from other Hyperion applications. Support for the earlier modules is unaffected by the new security platform. Implementation of the security platform is optional, but Hyperion strongly encourages using the security-platform authentication over the built-in modules authentication.

Managing Global Security for Applications and Databases

In addition to granting permissions to users and groups, you can change security settings for entire applications and databases and their related files and resources. Application and database security settings enable you to manage connections and create a lowest-common-security profile for the applications and databases.

This section contains the following sections:

Defining Application Settings

You can define permissions and other security settings that apply to applications by changing the application settings. The settings you define for the application affect all users, unless they have higher permissions granted to them at the user level.

Only users with Supervisor permission (or Application Designer permission for the application) can change application settings.

To define settings for an application, see "Setting General Application Connection Options" on page 850 or "Setting Application and Database Minimum Permissions" on page 854.

Setting General Application Connection Options

The following application settings are available:

- A brief description of the application
- A time-out setting for locks
- Options that control application loading, command processing, connections, updates, and security
- Settings that define the minimum permissions to the application. For detailed information about minimum permissions, see "Setting Application and Database Minimum Permissions" on page 854.

The following settings are available for various levels of application security. For information about how and when disabling these settings takes effect, see Table 30.

- Allow Application to Start

  When disabled, prevents all users from starting the application directly or as a result of operations that would start the application; for example, attempting to change application settings or create databases. By default, the application is not prevented from starting.

- Start When Analytic Server Starts

  When enabled, the application starts automatically whenever the Analytic Server starts. By default, the application does not start when the Analytic Server starts.

- Allow commands

  When unchecked, prevents any user from making any requests to databases in the application, including non-data-specific requests such as viewing database information or changing database settings. Supervisors are affected by this setting as a safety mechanism to prevent accidental changes to databases during maintenance operations. By default, commands are enabled.

- Allow connects

  When unchecked, prevents any user with a permission lower than Application Designer for that application from making connections to databases within the application which require the databases to be started. By default, connections to databases are allowed.

- Allow updates

  When unchecked, prevents modification to on-disk database structures; for example, any operation that might have an effect on the data, including updating the data or making outline modifications. Supervisors are affected by this setting as a safety mechanism to prevent accidental updates to databases during maintenance operations. By default, updates are enabled.

- Enable Security

  When unchecked, Analytic Services ignores all security settings in the application and treats all users as Application Designers. By default, security is enabled.

Table 30 describes when the implementation of protective application settings takes effect, how long the effects last, and which users are affected.

Table 30: Scope and Persistence of Application-Protection Settings

Disabled Application Setting	When the Disabled Setting Takes Effect	Which Users are Affected by the Disabled Setting	Persistence of the Disabled Setting
Allow Users to Start Application	Immediately	All users, including supervisors. Users currently logged on and users who log on later.	The application cannot be started until an administrator re-enables the start-up setting.
Start Application When Analytic Server Starts	Immediately	All users.	The application will not start with Analytic Server unless an administrator enables it.
Allow Commands	Immediately	All users, including supervisors. Users currently logged on and users who log on later.	Commands are disabled until any of the following actions occur: 1. The administrator who disabled commands logs off. 2. The application is stopped and restarted. 3. An administrator re-enables commands.
Allow Connects	Immediately, except that disabling connections does not affect users who already have databases loaded.	Users with permissions lower than Application Designer. Users currently logged on and users who log on later. Users already connected to the database are not affected.	Connections are disabled until any of the following actions occurs: 1. The application is stopped and restarted. 2. An administrator re-enables connections.

Table 30: Scope and Persistence of Application-Protection Settings (Continued)

Disabled Application Setting	When the Disabled Setting Takes Effect	Which Users are Affected by the Disabled Setting	Persistence of the Disabled Setting
Allow Updates	Immediately	All users, including supervisors. Users currently logged on and users who log on later.	Updates are disabled until any of the following occurs actions: 1. The administrator who disabled updates logs off. 2. The application is stopped and restarted. 3. An administrator re-enables updates.
Enable Security	Immediately	All users, including supervisors. Users currently logged on and users who log on later.	Security is disabled until a user re-enables security.

➤ To change application settings, use either of the following methods:

Tool	Topic	Location
Administration Services	Setting Application Properties	*Essbase Administration Services Online Help*
MaxL	**alter application**	*Technical Reference*

Note: If performing maintenance operations that require disabling *commands* or *updates*, make those maintenance operations within the same session as the one in which the setting was disabled.

If you disable commands or updates in a MaxL script, be aware that the end of the script constitutes the end of the session. Calling a nested MaxL or ESSCMD script from the current MaxL script also constitutes the end of the session.

If you disable commands or updates in an ESSCMD script, the end of the script constitutes the end of the session, but calling a nested ESSCMD script from the current ESSCMD script does *not* constitute the end of the session.

CAUTION: *Never* power down or reboot your client computer when you have cleared any of the Allow settings. Always log off from the server correctly. Improper shutdown can cause the application to become inaccessible, which requires a full application shutdown and restart.

If a power failure or system problem causes Analytic Server to improperly disconnect from the Analytic Services client, and the application is no longer accessible, you must shut down and restart the application. For instructions about starting and stopping Analytic Services, see "Starting and Stopping Analytic Server" on page 917.

Setting Application and Database Minimum Permissions

Minimum database access permissions can be specified at the application or database level. If specified for an application, minimum database access permissions apply to all databases within the application. When a minimum permission is set to a level higher than None (or No Access) for an application or database, all users inherit that permission to access the database or databases.

For example, if an application has Read permission assigned as the minimum database access level, all users can read any database within that application, even if their individual permissions do not include Read access. Similarly, if a database has a minimum permission setting of None, only users with sufficient granted permissions (granted directly, or implied by filters or group membership) can gain access to the database.

Users with Supervisor, Application Designer, or Database Designer permissions are not affected by minimum-permission settings applied to applications or databases they own. Supervisors have full access to all resources, and Application Designers and Database Designers have full access for their applications or databases.

Users and groups with lower than the minimum permissions inherit at least the minimum permissions for any applications or databases.

Changes to the minimum permission settings for applications affect only those databases that have lower minimums. In other words, settings defined at a lower level take precedence over more global settings.

The permissions listed in Table 31 are available as minimum settings for applications and databases. Databases of an application inherit the permissions of the applications whenever the application permissions are set higher than those of the database.

Table 31: Minimum Permission Settings Available for Applications and Databases

Permission	Description
None	Specifies that no minimum permission has been defined for the application or database. None is the default global permission for newly created applications and databases.
Read	Specifies Read-Only access to any object or data value in the application or database. Users can view files, retrieve data values, and run report scripts. Read access does not permit data-value updates, calculations, or outline modifications.
Write	Specifies Update access to any data value in the databases of the application, or in one database. Users can view Analytic Services files, retrieve and update data values, and run report scripts. Write access does not permit calculations or outline modifications.
MetaRead	Gives Read access to the specified members, but hides data for their ancestors, and hides data and metadata for their siblings.
Calculate	Specifies Calculate and update access to any data value in the databases of the application, or in one database. Users can view files, retrieve, update, and perform calculations based on data values, and run report and calculations scripts. Calculate access does not permit outline modifications.
Designer (for Application or Database)	Specifies Calculate and update access to any data value in the databases of the application, or in one database. In addition, Designer permission enables users to view and modify the outline and files, retrieve, update, and perform calculations based on data values, and run report and calculation scripts.

Note: Although any user with a minimum of Read access to a database can start the database, only a Supervisor, a user with Application Designer permission for the application, or a user with Database Designer permission for the database can stop the database.

- To set minimum permissions for an application, see "Setting Minimum Permissions for Applications" in the *Essbase Administration Services Online Help*, or see **alter application** in the MaxL DDL section of the *Technical Reference*.

- To set minimum permissions for a database, see "Setting Minimum Permissions for Databases" in the *Essbase Administration Services Online Help*, or see **alter database** in the MaxL DDL section of the *Technical Reference*.

Managing User Activity on the Analytic Server

This topic explains how to manage the activities of users connected to the Analytic Server. The security concepts explained in this section are session and request management, lock management, connection management, and password and user name management. For information about managing security for partitioned databases, see Chapter 13, "Designing Partitioned Applications."

Disconnecting Users and Terminating Requests

The security system lets you disconnect a user from the Analytic Server when you want to perform maintenance tasks.

To view sessions, disconnect sessions, or terminate requests, you must have Supervisor permission or Application Designer permission for the specified application. You can view or terminate only sessions or requests for users with permissions equal to or lesser than your own.

A *session* is the time between login and logout for a user connected to Analytic Server at the system, application, or database scope. A user can have more than one session open at any given time. For example, a user may be logged on to different databases. If you have the appropriate permissions, you can log off sessions based on any criteria you choose; for example, an administrator can log off a user from all databases or from a particular database.

A *request* is a query sent to Analytic Server by a user or by another process; for example, a default calculation of a database, or a restructuring of the database outline. Each session can process only one request at a time.

Note: You cannot terminate a restructure process. If you attempt to terminate it, a "command not accepted" error is returned, and the restructure process is not terminated.

- To disconnect a session or request using Administration Services, see "Disconnecting User Sessions and Requests" in the *Essbase Administration Services Online Help*.
- To disconnect a session or request using MaxL, see the following table:

Table 32: Session and Request Termination Options

Selections in Administration Services			MaxL equivalents
log off	selected user	only	`alter system logout session <session-id>;`
log off	all users	on selected server	`alter system logout session all;`
log off	all users	on selected application	`alter system logout session on application <app-name>;`
log off	all users	on selected database	`alter system logout session on database <dbs-name>;`
log off	all instances of user	on selected server	`alter system logout session by user <user-name>;`
log off	all instances of user	on selected application	`alter system logout session by user <user-name> on application <app-name>;`
log off	all instances of user	on selected database	`alter system logout session on database <dbs-name>;`
kill	selected request	only	`alter system kill request <session-id>;`
kill	all requests	on selected server	`alter system kill request all;`

Table 32: Session and Request Termination Options (Continued)

Selections in Administration Services			MaxL equivalents
kill	all requests	on selected application	`alter system kill request on application <app-name>;`
kill	all requests	on selected database	`alter system kill request on database <dbs-name>;`
kill	all requests from user	on selected server	`alter system kill request by user <user-name>;`
kill	all requests from user	on selected application	`alter system kill request by user <user-name> on application <app-name>;`
kill	all requests from user	on selected database	`alter system kill request by user <user-name> on database <dbs-name>;`

Managing User Locks

Spreadsheet Add-in users can interactively send data from a spreadsheet to the server. To maintain data integrity while providing multiple-user concurrent access, Analytic Services enables users to lock data for the purpose of updating it. Users who want to update data must first lock the records to prevent other users from trying to change the same data.

Occasionally, you may need to force an unlock operation. For example, if you attempt to calculate a database that has active locks, the calculation must wait when it encounters a lock. By clearing the locks, you allow the calculation to resume.

Only Supervisors can view users holding locks and remove their locks.

➤ To view or remove locks, use either of the following methods:

Tool	Topic	Location
Administration Services	Viewing Data Locks and Unlocking Data	*Essbase Administration Services Online Help*
MaxL	**drop lock**	*Technical Reference*

Managing Passwords and User Names

You can place limitations on the number of login attempts users are allowed, on the number of days users may not use Analytic Services before becoming disabled from the server, and on the number of days users are allowed to have the same passwords. Only system administrators (users with Supervisor permission) can access these settings. The limitations apply to all users on the server, and are effective immediately upon clicking OK.

Note: If you later change the number of unsuccessful login attempts allowed, Analytic Services resets the count for all users. For example, if the setting was 15 and you changed it to 20, all users are allowed 20 *new* attempts. If you changed the setting to 2, a user who had already exceeded that number when the setting was 15 is *not* locked out. The count returns to 0 for each change in settings.

➤ To place limitations on users, use any of the following methods:

Tool	Topic	Location
Administration Services	Managing Password Longevity Disconnecting Users Automatically	*Essbase Administration Services Online Help*
MaxL	**alter system**	*Technical Reference*
MaxL	**alter user**	*Technical Reference*

Propagating Password Changes

You can change a user's password and then propagate the new password to other Analytic Servers. You need Create/Delete Users and Groups permissions for both the source and the target servers. The user whose password you are changing must exist on the target servers, and the target servers must be running.

If you use Administration Services to change a user's Analytic Server password, and if the user is also an Administration Services user, the user's Administration Services user properties are updated automatically. The user's Administration Services password is not affected. For more information, see "Changing Passwords for Analytic Server Users" in *Essbase Administration Services Online Help*.

➤ To change a user's Analytic Server password and propagate the new password to other Analytic Servers, see "Propagating Passwords Across Servers" in *Essbase Administration Services Online Help*.

Viewing and Activating Disabled User Names

You can prevent a user from logging in to an Analytic Server by disabling the user name at the Analytic Server level. A username is disabled automatically when the user exceeds limitations specified, or a username can be disabled manually for individual users. For more information about limitations that cause user names to become disabled automatically, see "Managing Passwords and User Names" on page 859.

Administration Services provides a Disabled Usernames window that enables you to view and activate all usernames that have been disabled for an Analytic Server. Only a user with at least Create/Delete User permission can view or reactivate disabled user names.

➤ To disable a user name manually, use either of the following methods:

Tool	Topic	Location
Administration Services	Disabling Usernames	*Essbase Administration Services Online Help*
MaxL	**alter user**	*Technical Reference*

➤ To view or activate currently disabled user names, use either of the following methods:

Tool	Topic	Location
Administration Services	Viewing or Activating Disabled Usernames	*Essbase Administration Services Online Help*
MaxL	**alter user**	*Technical Reference*

Understanding the essbase.sec Security File and Backup

All information about users, groups, passwords, permissions, filters, applications, databases, and their corresponding directories is stored in the `essbase.sec` file in the *ARBORPATH*`\bin` directory. Each time you successfully start the Analytic Server, a backup copy of the security file is created as `essbase.bak`. You can also update the security backup file more often by using one of the following methods:

- Manually compare the security backup file to the security file at any time and update the backup file if necessary
- Specify an interval at which Analytic Services automatically compares the security backup file to the security file and updates the backup file if necessary

➤ To update the security backup file, use either of the following methods:

Tool	Topic	Location
Administration Services	Updating the Security Backup File	*Essbase Administration Services Online Help*
MaxL	**alter system sync security backup**	*Technical Reference*

If you attempt to start Analytic Server and cannot get a password prompt or your password is rejected, no `.bak` file is created. You can restore from the last successful startup by copying `essbase.bak` to `essbase.sec`. Both files are in the `essbase\bin` directory where you installed Analytic Services.

CAUTION: If Analytic Services stops running unexpectedly for any reason, such as a freeze or crash, or as the result of terminating a process, do not restart Analytic Server until you copy the backup file (`essbase.bak`) to the security file (`essbase.sec`). If you do not perform the copy first, when Analytic Server starts, Analytic Services notes that `essbase.sec` is corrupt, creates an empty security file, and copies it to `essbase.bak`, thus destroying the backup of your security information.

Chapter 37

Controlling Access to Database Cells

When the security levels defined for applications, databases, users, and groups are not enough, Analytic Services security filters give you control over security at the most detailed level. Filters let you control access to individual data within a database, by defining what kind of access is allowed to which parts of the database, and to whom these settings apply.

If you have Supervisor permissions, you can define and assign any filters to any users or groups. Filters do not affect you.

If you have Create/Delete Applications permissions, you can assign and define filters for applications you created.

If you have Application Designer or Database Designer permissions, you can define and assign filters within your applications or databases. For descriptions of permission levels, see "Understanding Security and Permissions" on page 836.

This chapter contains the following sections:

Understanding How Filters Define Permissions

Filters control security access to data values, or cells. You create filters to accommodate security needs for specific parts of a database. When you define a filter, you are designating a set of restrictions upon particular database cells. When you save the filter, you give it a unique name to distinguish it from other filters, and the server stores it in `ESSBASE.SEC`, the security file. You can then assign the filters to any users or groups on the server.

For example, a manager designs a filter named RED, and associates it with a particular database to limit access to cells containing profit information. The filter is assigned to a visiting group called REVIEWERS, so that they can read, but cannot alter, most of the database, while they have no access at all to Profit data values.

Filters are composed of one or more access settings for database members. You can specify the following access levels and apply them to data ranging from a list of members to an individual cell.

Access Level	Description
None	No data can be retrieved or updated for the specified member list.
Read	Data can be retrieved but not updated for the specified member list.
Write	Data can be retrieved and updated for the specified member list.
MetaRead	Metadata (dimension and member names) can be retrieved and updated for the corresponding member specification.

Note:

The MetaRead access level overrides all other access levels. If additional filters for data are defined, they are enforced within any defined MetaRead filters.

If you have assigned a MetaRead filter on a substitution variable, then try to retrieve the substitution variable, an unknown member error occurs, but the value of the substitution variable gets displayed. This is expected behavior.

Metadata security cannot be completely turned off in partitions. Therefore, do not set metadata security at the source database; otherwise, incorrect data may result at the target partition.

When drilling up or retrieving on a member that has metadata security turned on and has shared members in the children, an unknown member error occurs because the original members of the shared members have been filtered. To avoid getting this error, be sure to give the original members of the shared members metadata security access.

Any cells that are not specified in the filter definition inherit the database access level. Filters can, however, add or remove access assigned at the database level, because the filter definition, being more data-specific, indicates a greater level of detail than the more general database access level.

Note: Data values not covered by filter definitions default first to the access levels defined for users and second to the global database access levels. For a detailed discussion of user access levels, see "Granting Permissions to Users and Groups" on page 840. For a detailed discussion of global access levels, see "Managing Global Security for Applications and Databases" on page 849.

Calculation access is controlled by minimum global permissions or by permissions granted to users and groups. Users who have calculate access to the database are not blocked by filters—they can affect all data elements that the execution of their calculations would update.

Creating Filters

You can create a filter for each set of access restrictions you need to place on database values. There is no need to create separate filters for users with the same access needs—once you have created a filter, you can assign it to multiple users or groups of users. However, only one filter per database can be assigned to a user or group.

Note: If you use a calculation function that returns a set of members, such as children or descendants, and it evaluates to an empty set, the security filter is not created. An error is written to the application log stating that the region definition evaluated to an empty set.

Before creating a filter perform the following actions:

- Connect to the server and select the database associated with the filter.
- Check the naming rules for filters in Appendix A, "Limits."

➤ To create a filter, use either of the following methods:

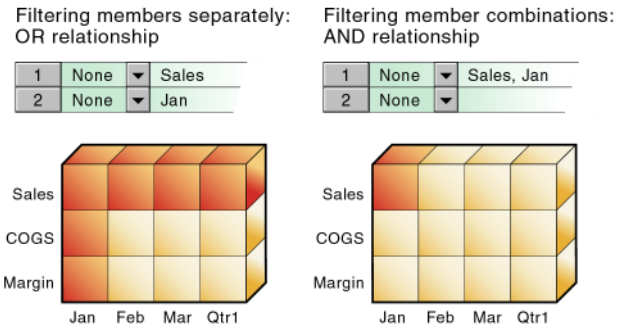
Tool	Topic	Location
Administration Services	Creating or Editing Filters	*Essbase Administration Services Online Help*
MaxL	**create filter**	*Technical Reference*

Filtering Members Versus Filtering Member Combinations

Figure 209 illustrate different ways to control access to database cells. Data can be protected by filtering entire members, or by filtering member combinations.

- Filtering members separately affects whole regions of data for those members.
- Filtering member combinations affects data at the member intersections.

Figure 209: How Filters Affect Data AND/OR Relationships



Note: Filtering on member combinations (AND relationship) does not apply to MetaRead. MetaRead filters each member separately (OR relationship).

Filtering Members Separately

To filter all the data for one or more members, define access for each member on its own row in Filter Editor. Filter definitions on separate rows of a filter are treated with an OR relationship.

For example, assume that user KSmith is assigned this filter:

Figure 210: Filter Blocking Access to Sales or Jan

Access	Member Specification
None	Sales
None	Jan

The filter blocks access to all members Sales or Jan in the Sample Basic database.

The next time user KSmith connects to Sample Basic, she has no access to data values for the member Sales or for the member Jan. Her spreadsheet view of the profit margin for Qtr1 looks like this view:

Figure 211: Results of Filter Blocking Access to Sales or Jan

	A	B	C	D	E
1				Product	Market
2	Sales	Jan	Scenario	#NoAccess	
3		Feb	Scenario	#NoAccess	
4		Mar	Scenario	#NoAccess	
5		Qtr1	Scenario	#NoAccess	
6	COGS	Jan	Scenario	#NoAccess	
7		Feb	Scenario	14307	
8		Mar	Scenario	14410	
9		Qtr1	Scenario	42877	
10	Margin	Jan	Scenario	#NoAccess	
11		Feb	Scenario	17762	
12		Mar	Scenario	17803	
13		Qtr1	Scenario	52943	

All data for Sales is blocked from view, as well as all data for January, inside and outside of the Sales member. Data for COGS (Cost of Goods Sold), a sibling of Sales and a child of Margin, is available, with the exception of COGS for January.

Filtering Member Combinations

To filter data for member combinations, define the access for each member combination using a single row in Filter Editor. A filter definition using one row and a comma is treated as an AND relationship.

For example, assume that user RChinn is assigned this filter:

Figure 212: Filter Blocking Access to Sales for Jan

Access	Member Specification
None	Sales, Jan

The filter blocks only the intersection of the members Sales and Jan in the Sample Basic database.

The next time user RChinn connects to Sample Basic, she has no access to the data value at the intersection of members Sales and Jan. Her spreadsheet view of the profit margin for Qtr1 looks like this view:

Figure 213: Results of Filter Blocking Access to Sales, Jan

	A	B	C	D	E
1			Product	Market	Scenario
2	Sales	Jan	#NoAccess		
3		Feb	32069		
4		Mar	32213		
5		Qtr1	95820		
6	COGS	Jan	14160		
7		Feb	14307		
8		Mar	14410		
9		Qtr1	42877		
10	Margin	Jan	17378		
11		Feb	17762		
12		Mar	17803		
13		Qtr1	52943		
14					

Sales data for January is blocked from view. However, Sales data for other months is available, and non-Sales data for January is available.

Filtering with Attribute Functions

You can use filters to restrict access to data for base members sharing a particular attribute. To filter data for members with particular attributes defined in an attribute dimension, use the attribute member in combination with the @ATTRIBUTE function or the @WITHATTR function.

Note: @ATTRIBUTE and @WITHATTR are member set functions. Most member set functions can be used in filter definitions.

For example, assume that user PJones is assigned this filter:

Figure 214: Filter Blocking Access to Members with Attribute "Caffeinated_False"

Access	Member Specification
None	@ATTRIBUTE("Caffeinated_False")

The filter blocks access to data for caffeine-free products. “Caffeinated_False” is a Boolean-type attribute member in Sample Basic, in the Pkg_Type attribute dimension. This attribute is associated with members in the Product base dimension.

The next time user PJones connects to Sample Basic, he has no access to the data values for any base dimension members associated with Caffeinated_False. His spreadsheet view of first-quarter cola sales in California looks like this view:

Figure 215: Results of Filter Blocking Access to Caffeine-free Products

	Sales	California	Qtr1	Actual
Cola	1998			
Diet Cola	367			
Caffeine Free Cola	#Miss			
Colas	2767			

Sales data for Caffeine Free Cola is blocked from view. Note that Caffeine Free Cola is a base member, and Caffeinated_False is an associated member of the attribute dimension Caffeinated (not shown in the above spreadsheet view).

Managing Filters

You can perform the following actions on filters:

- “Viewing Filters” on page 870
- “Editing Filters” on page 870
- “Copying Filters” on page 870
- “Renaming Filters” on page 871
- “Deleting Filters” on page 871

Viewing Filters

➤ To view a list of filters, use any of the following methods:

Tool	Topic	Location
Administration Services	Creating or Editing Filters	*Essbase Administration Services Online Help*
MaxL	**display filter**	*Technical Reference*
ESSCMD	LISTFILTERS	*Technical Reference*

Editing Filters

➤ To edit an existing filter, use either of the following methods:

Tool	Topic	Location
Administration Services	Creating or Editing Filters	*Essbase Administration Services Online Help*
MaxL	**create filter**	*Technical Reference*

Copying Filters

You can copy filters to applications and databases on any Analytic Server, according to your permissions. You can also copy filters across servers as part of application migration.

➤ To copy an existing filter, use any of the following methods:

Tool	Topic	Location
Administration Services	Copying Filters	*Essbase Administration Services Online Help*
MaxL	**create filter**	*Technical Reference*
ESSCMD	COPYFILTER	*Technical Reference*

Renaming Filters

➤ To rename an existing filter, use any of the following methods:

Tool	Topic	Location
Administration Services	Renaming Filters	*Essbase Administration Services Online Help*
MaxL	**create filter**	*Technical Reference*
ESSCMD	RENAMEFILTER	*Technical Reference*

Deleting Filters

➤ To delete an existing filter, use either of the following methods:

Tool	Topic	Location
Administration Services	Deleting Filters	*Essbase Administration Services Online Help*
MaxL	**drop filter**	*Technical Reference*

Assigning Filters

Once you have defined filters, you can assign them to users or groups. Assigning filters to users and groups lets you manage multiple users who require the same filter settings. Modifications to the definition of a filter are automatically inherited by users of that filter.

Filters do not affect users who have the role of Supervisor. Only one filter per database can be assigned to a user or group.

➤ To assign a filter to a user or group, see “Assigning Filters” in the *Essbase Administration Services Online Help*.

Overlapping Filter Definitions

If a filter contains rows that have overlapping member specifications, the inherited access is set by the following rules, which are listed in order of precedence:

1. A filter that defines a more detailed dimension combination list takes precedence over a filter with less detail.
2. If the preceding rule does not resolve the overlap conflict, the highest access level among overlapping filter rows is applied.

For example, this filter contains overlap conflicts:

Table 33: Filter with Overlap Conflicts

Access	Member Specification
Write	Actual
None	Actual
Read	Actual, @IDESCENDANTS(“New York”)

The third specification defines security at a greater level of detail than the other two. Therefore Read access is granted to all Actual data for members in the New York branch.

Because Write access is a higher access level than None, the remaining data values in Actual are granted Write access.

All other data points, such as Budget, are accessible according to the minimum database permissions.

Note: If you have Write access, you also have Read access.

Changes to members in the database outline are not reflected automatically in filters. You must manually update member references that change.

Overlapping Access Definitions

When the access rights of user and group definitions overlap, the following rules, listed in order of precedence, apply:

1. An access level that defines a more detailed dimension combination list takes precedence over a level with less detail.
2. If the preceding rule does not resolve the overlap conflict, the highest access level is applied.

Example 1:

User Fred is defined with the following database access:

```
FINPLAN     R
CAPPLAN     W
PRODPLAN    N
```

He is assigned to Group Marketing which has the following database access:

```
FINPLAN     N
CAPPLAN     N
PRODPLAN    W
```

His effective rights are set as follows:

```
FINPLAN     R
CAPPLAN     W
PRODPLAN    W
```

Example 2:

User Mary is defined with the following database access:

```
FINPLAN     R
PRODPLAN    N
```

She is assigned to Group Marketing which has the following database access:

```
FINPLAN     N
PRODPLAN    W
```

Her effective rights are set as follows:

```
FINPLAN     R
PRODPLAN    W
```

In addition, Mary uses the filter object RED (for the database FINPLAN). The filter has two filter rows:

Figure 216: RED Filter for Database FINPLAN

Access	Member Specification
Read	Actual
Write	Budget, @IDESCENDANTS(“New York”)

The Group Marketing also uses a filter object BLUE (for the database FINPLAN). The filter has two filter rows:

Figure 217: BLUE Filter for Database FINPLAN

Access	Member Specification
Read	Actual, Sales
Write	Budget, Sales

Mary’s effective rights from the overlapping filters, and the permissions assigned to her and her group, are as follows:

R Entire Finplan database

W For all Budget data in the New York branch

W For data values that relate to Budget and Sales

For additional sample scenarios, see Chapter 38, “Security Examples.”

Chapter 38

Security Examples

This chapter describes some sample security problems and solutions, which are based on the Sample application. These examples use security procedures described in Chapter 36, "Managing Security for Users and Applications."

Security Problem 1

Three employees need to use Analytic Services—Sue Smith, Bill Brown, and Jane Jones. Each requires update access to all databases in the Sample application.

Solution:

Because the users need update access to only one application, they do not need to have Supervisor permission. Because the users do not need to create or delete applications, users, or groups, they do not need to be defined as special types of users with Create/Delete permission. All these users need is Application Designer permission for the Sample application.

The supervisor should perform the following tasks:

1. Set up the users with Administration Services.

   For more information, see *Essbase Administration Services Installation Guide*.

2. Create Sue, Bill, and Jane as ordinary users with Application Designer permission.

   If Sue, Bill, and Jane are created without Application Designer permission, assign Application Designer permission to the three users.

   For more information, see "Creating Users" on page 839 or "Granting Designer Permissions to Users and Groups" on page 844.

Security Problem 2

Three employees need to use Analytic Services—Sue Smith, Bill Brown, and Jane Jones. Sue and Bill require full access to all databases in the Sample application. Jane requires full calculate access to all databases in the Sample application, but she does not need to define or maintain database definitions.

Solution:

The supervisor should perform the following tasks:

1. Set up the users with Administration Services.

   See the *Essbase Administration Services Installation Guide*.

2. Create Sue and Bill as ordinary users with Application Designer permission.

   If Sue and Bill are created without Application Designer permission, assign Application Designer permission to the two users.

   For more information, see "Creating Users" on page 839 or "Granting Designer Permissions to Users and Groups" on page 844.

3. Define global Calculate access for the Sample application as the Minimum Database Access setting to give all additional users Calculate access to all databases for the application.

   See "Setting Minimum Permissions for Applications" in the *Essbase Administration Services Online Help*.

4. Create Jane as an ordinary user with no additional permissions. She inherits the Calculate access from the application global setting.

   For more information, see "Creating Users" on page 839.

Security Problem 3

Three employees need to use Analytic Services—Sue Smith, Bill Brown, and Jane Jones. Sue and Bill require full access to all databases in the Sample application. Jane requires full update and calculate access to all databases within the Sample application, but she will not define or maintain the database definitions. Additional users will be added, all of whom will require Read access to all databases.

Solution:

Because the current users have different needs for application and database access, define their user permissions individually. Then, to save time assigning individual Read permissions for future users, make Read the global setting for the application. (It does not matter in what order you assign the user permissions and the global access.)

The supervisor should perform the following tasks:

1. Set up the users with Administration Services.

   For more information, see *Essbase Administration Services Installation Guide*.

2. Create or edit Sue and Bill as ordinary users with Application Designer permissions.

   For more information, see "Creating Users" on page 839 and "Granting Designer Permissions to Users and Groups" on page 844.

3. Create Jane as an ordinary user, and give her Calculate permission for the Sample application.

   For more information, see "Creating Users" on page 839 and "Granting Application and Database Access to Users and Groups" on page 842.

4. Define global Read access for the Sample application as the Minimum Database Access setting to give all additional users Read access to all databases in the Sample application.

   See "Setting Minimum Permissions for Databases" in the *Essbase Administration Services Online Help*.

Security Problem 4

Three employees need to use Analytic Services—Sue Smith, Bill Brown, and Jane Jones. Sue requires full access only to the Sample application; Jane requires calculate access to all members of the Basic database; Bill requires Read access to all members. No other users should have access to the databases.

Furthermore, Jane and Bill need to run report scripts that are defined by Sue.

Solution:

Because the different users have different needs for application and database access, define the global access setting as None, and assign the user permissions individually.

The supervisor should perform the following tasks:

1. Set up the users with Administration Services. (Because Jane and Bill need to run the report scripts, they must use Administration Services.)

   For more information, see *Essbase Administration Services Installation Guide*.

2. Create Sue as an ordinary user, but grant her Application Designer permission for the Sample application.

   For more information, see "Creating Users" on page 839 and "Granting Designer Permissions to Users and Groups" on page 844.

3. Create Jane as an ordinary user, and give her Calculate permission for the Sample application.

   For more information, see "Creating Users" on page 839 and "Granting Application and Database Access to Users and Groups" on page 842.

4. Create Bill as an ordinary user and give him Read permission on the Sample application.

   For more information, see "Creating Users" on page 839 and "Granting Application and Database Access to Users and Groups" on page 842.

Security Problem 5

The Supervisor, Sue Smith, needs to perform some maintenance on the Sample application. She must make changes to the database outline and reload actual data. While she changes the application, Sue must prevent other users from connecting to the application.

Solution:

Sue should perform the following tasks:

1. Disable the Allow Commands setting to prevent other users from connecting to the application, and also prevent connected users from performing any further operations.

   For more information, see “Clearing Applications of User Activity” in the *Essbase Administration Services Online Help*.

2. Check to see if any users have active locks.

   If any users have active locks, Sue’s calculation or data load command might halt, waiting for access to the locked records. Sue can allow the users to complete their updates or clear their locks.

   For more information, see “Viewing Data Locks” in the *Essbase Administration Services Online Help*.

3. After confirming that no users have active locks, proceed to perform maintenance on the application.

Part

VII

Enabling Multi-Language Applications Through Unicode

This part describes how, through Unicode Standard, Essbase Analytic Services supports applications with character sets from multiple languages. This part also describes how to work with Unicode-mode applications:

- Chapter 39, “Analytic Services Implementation of Unicode” describes how, through Unicode, Analytic Services enables applications to be shared by users of computers with different languages.
- Chapter 40, “Administering Unicode-Mode Applications” describes how to set up and use Unicode-mode applications.

Chapter 39

Analytic Services Implementation of Unicode

Through its Unicode implementation, Essbase Analytic Services enables employees of global businesses to view, in their own languages, company information stored in their Analytic Services databases.

The following topics of this chapter provide a brief overview of Unicode and describe the Analytic Services implementation of Unicode Standard.

About Unicode

Sharing data across national and language boundaries is a challenge for multi-national businesses. Traditionally, each computer stores and renders text based on its locale specification. A *locale* identifies the local language and identifies cultural conventions, such as currency and date format, data sort order, and character set encoding. The *encoding* of a character set refers to the specific set of bit combinations used to store the character text as data, as defined by a code page or an encoding format. In Analytic Services, *code pages* map characters to bit combinations for non-Unicode encodings.

Because different encodings can map the same bit combination to different characters, a file created on one computer can be misinterpreted by another computer that has a different locale.

Unicode Standard was developed to enable computers with different locales to share character data. Unicode provides encoding forms with thousands of bit combinations, enough to support the character sets of multiple languages

simultaneously. By combining all character mappings into a single encoding form, Unicode enables users to view character data created on computers with different locale settings correctly.

Analytic Services conforms to version 2.1 of Unicode Standard and uses the popular UTF-8 encoding form within Unicode Standard. For additional information about the Unicode Standard, see `www.unicode.org`.

About the Analytic Services Implementation of Unicode

By providing Unicode support, Analytic Services helps solve the multi-national business problem of sharing data across the enterprise. Analytic Services users whose computers are set up in different languages can work with the same database. For example, using alias tables in their respective languages, users in Taiwan can view database reports in Chinese characters while users in France can view the same reports in French characters.

The following topics describe the fundamental characteristics of the Analytic Services implementation of Unicode:

Note: User-defined character sets (UDC) are not supported. SQL Interface does not work with Unicode-mode applications.

Locales and Analytic Services

Analytic Services uses the ESSLANG variable to define the locale of a computer. You must specify the ESSLANG variable for each Analytic Server installation, and you should set ESSLANG to the locale that is defined for the operating system of the computer. Defining an ESSLANG variable on client computers is optional. If ESSLANG is defined on a client, Analytic Services uses the ESSLANG value as the computer locale. If no ESSLANG value is specified on a client, Analytic Services uses the locale specified for the operating system. For information about defining the ESSLANG variable, see the *Essbase Analytic Services Installation Guide*.

Note: As described in "About Unicode" on page 883, a locale specification contains several kinds of information about the language and culture that the computer supports. Analytic Services uses only the code-page portion of locale specifications. The cultural conventions and sort-order portions are not used.

Unicode and Non-Unicode Application Modes

Applications are designated as Unicode-mode applications or non-Unicode-mode applications.

Unicode-mode applications support multiple character sets. When Analytic Services works with Unicode-mode applications, it uses the UTF-8 encoding form to interpret and store character text. Character-based objects in Unicode-mode applications, such as member and alias names, can include characters from different languages.

Because Unicode-mode applications accept input files in non-Unicode-encoding as well as in UTF-8, Analytic Services relies on locale indicators and user prompting to read or write non-Unicode-encoded files. For a basic description of encoding, see "About Unicode" on page 883.

Clients working with Unicode-mode applications can have locales different from the locale of Analytic Server. For example, client computers with Japanese locales and client computers with German locales can work with the same Unicode-mode application on an instance of Analytic Server that has a Spanish locale.

For Unicode-mode applications, the limits of most object names are longer than the limits in non-Unicode-mode applications, and the limits are calculated based on characters rather than bytes. See "Increased Name Lengths" on page 886.

Non-Unicode-mode applications support one character set that is defined by a locale value that must be the same for Analytic Server and all non-Unicode clients that work with non-Unicode-mode applications. By default, Analytic Services creates applications in non-Unicode mode.

You can define an application as Unicode-mode when you create the application, or you can migrate a non-Unicode-mode application to Unicode mode in a separate step. For information about these processes, see "Creating a New Unicode-Mode Application" on page 896.

Note: You cannot convert a Unicode-mode application to non-Unicode mode.

Both Unicode-mode and non-Unicode-mode applications can reside on the same Analytic Server. For information to help determine the type of application to use for particular situations, see "When to Use Unicode-Mode Applications" on page 891.

Unicode and Non-Unicode Analytic Server Modes

Analytic Server must have permission to create Unicode-mode applications and to migrate existing applications to Unicode mode. When this permission is enabled, Analytic Server is said to be in Unicode mode. For information about granting or revoking this permission, see "Setting Analytic Server to Unicode Mode" on page 895.

Whether or not Analytic Server is in Unicode mode affects only the creation and migration of applications. Regardless of the Analytic Server Unicode setting, you can work with both Unicode mode and non-Unicode-mode applications.

Increased Name Lengths

For Unicode-mode applications, the maximum number of characters allowed in strings, such as application, database, and member names, is greater than the maximum allowed for non-Unicode-mode applications.

For non-Unicode-mode applications, the maximum length of most object names is calculated in bytes. For Unicode-mode applications, the maximum length of most object names is calculated based on the number of characters, regardless of how many bytes each character requires. Not limiting by bytes is advantageous for applications using multi-byte character sets, such as Chinese and Japanese. For

example, member names in Unicode-mode applications can contain 80 characters, even if they are multi-byte characters. For a list of object-name size maximums for Unicode-mode and non-Unicode-mode applications, see Appendix A, "Limits."

Note: The increase in size limits does not affect the size of the outline, but may affect the size of user-written client programs.

To take advantage of longer name sizes, users may decide to work with Unicode-mode applications, even if all users work in the same locale (See "When to Use Unicode-Mode Applications" on page 891.).

Compatibility Between Different Versions of Client and Server Software

To support customers as they upgrade their computers to a Unicode-enabled release of Analytic Services, Analytic Services enables different versions of client and server software to communicate with each other. Table 34 summarizes the compatibility between different versions of Analytic Services client and server software. The following terminology is used in the table:

- Not Unicode-enabled Analytic Server: Analytic Server installed using a release of Analytic Services prior to Release 7.0.
- Unicode-enabled Analytic Server: Analytic Server installed using Analytic Services Release 7.0 or later.
- Not Unicode-enabled client: a client program built using include files and DLLs from non-Unicode releases of Analytic Services. Clients that are not Unicode enabled deal entirely in short strings and non-Unicode encoding. Application Manager is an example of a client that is not Unicode enabled.
- Unicode-enabled client: a client program built using include files and DLLs from Unicode-enabled releases of Analytic Services. Unicode-enabled client programs can be in Unicode mode or non-Unicode mode.
  - Unicode-mode client program: a Unicode-enabled client program that communicates with Analytic Services API in Unicode encoding, using long strings (See the *API Reference* for details.). Administration Services Console is an example of a Unicode-mode client.
  - Non-Unicode-mode client program: a Unicode-enabled client program that communicates with Analytic Services API in short strings (See the *API Reference* for details.). Excel Spreadsheet Add-in is an example of a non-Unicode-mode client.

- Unicode-mode application: an Analytic Services application that uses UTF-8 encoding to display and store character text.
- Non-Unicode-mode application: an Analytic Services application that displays and stores character text based on the locale common to clients and server.

Table 34: Compatibility Between Different Versions of Clients and Analytic Server

	Not Unicode-enabled Analytic Server	**Unicode-enabled Analytic Server, Non-Unicode-mode application**	**Unicode-enabled Analytic Server, Unicode-mode application**
Not Unicode-enabled client For example: Application Manager and Essbase Spreadsheet Add-in releases prior to 7.0	Yes	Yes	No
Non-Unicode-mode client program on a Unicode-enabled client For example: MaxL Shell and Essbase Excel Spreadsheet Add-in packaged with Unicode-enabled Analytic Services	Yes, except for clients using the grid API	Yes	Yes, excepting that no changes can be made to outlines No outline synchronization
Unicode-mode client program on a Unicode-enabled client Examples: Essbase Administration Services and Essbase Spreadsheet Services	Yes, except for clients using the grid API Synchronization of outlines not supported for applications encoded to locales with multi-byte characters	Yes. Synchronization of outlines not supported for applications encoded to locales with multi-byte characters	Yes

Support of Different Locales on Clients and Analytic Server

For each Analytic Server installation you must define the ESSLANG variable. For details about defining ESSLANG, see the *Essbase Analytic Services Installation Guide*. The ESSLANG variable is a *locale* definition, including a code page specification that maps bit combinations to characters. For example, to support American English, you can set ESSLANG to `English_UnitedStates.Latin1@Binary`.

Clients also use locales. ESSLANG variables are optional for clients. If ESSLANG is not defined on a client, the locale specified for the operating system is used.

Client and Analytic Server locale values must be the same for non-Unicode-mode clients and applications. For Unicode-mode applications, client and Analytic Server locale values can be different.

Identification of Text Encoding and Locale

Analytic Services also supports use of non-Unicode-encoded files with Unicode-mode applications. In order to identify what type of encoding a file is, Analytic Services looks for UTF-8 signatures or locale indicators to identify the encoding. If a file contains neither encoding identifier and the file is not on Analytic Server, Administration Services prompts the user for the locale of the file. For details about locale indicators, see "Encoding Indicators" on page 901.

The Analytic Services Unicode File Utility program (ESSUTF8) includes options to insert UTF-8 signatures or locale headers in text files or you can a text editor or other means to insert them. For general information about the utility, see "Analytic Services Unicode File Utility" on page 906. For description and syntax of this utility, see the *Technical Reference*.

Unicode-Enabled Administration Tools

Hyperion Solutions provides Essbase Administration Services and MaxL to administer Unicode-mode applications. The main administration activities include, in addition to the normal Analytic Services administration activities, changing the Unicode-related mode of Analytic Server to enable or disable the following abilities:

- Creation of Unicode-mode applications
- Migration of non-Unicode-mode applications to Unicode mode
- Viewing of Unicode-related status of Analytic Server and applications

Administration Services Console is a Unicode-mode client. You can use Administration Services Console with both Unicode and non-Unicode-mode applications. See Chapter 40, “Administering Unicode-Mode Applications” for information about Unicode-related administration tasks.

To administer non-Unicode-mode applications, you can use Application Manager from previous Analytic Services releases that are not Unicode enabled.

Unicode-Enabled C API

Without recompilation, custom-written client programs used with Analytic Services releases prior to Rel. 7.0 are not Unicode-enabled. Such programs use short strings and short buffers. You can continue to use these programs with non-Unicode-mode applications.

In order to provide restricted access to Unicode-mode applications, these older custom-written client programs, depending on how they are written, can be recompiled in a Unicode-enabled release of Analytic Services. When recompiled, the programs work with long buffers but short strings.

For complete access to Unicode-mode and non-Unicode-mode applications, existing custom-written applications need to be modified using the new Analytic Services API functions for Unicode. Rewritten and compiled clients work with long buffers and long strings for full Unicode support. For information about updating custom-written client programs, see the *API Reference*.

Spreadsheet Retrieval

You can use Spreadsheet Services to view data in both Unicode-mode applications and non-Unicode-mode applications. To run Spreadsheet Services, you must also run Essbase Deployment Services. See the installation guides for each of these products for preparation and installation information The Spreadsheet Add-in does not support Unicode.

Sample_U Basic

To demonstrate Unicode-mode applications, the sample applications include a Unicode-mode application and database: Sample_U Basic. Member names in Sample_U Basic are in English.

Sample_U Basic includes four non-English alias tables and their import files: `nameschn.alt` (Chinese), `namesger.alt` (German), `namesjpn.alt` (Japanese), and `namesrsn.alt` (Russian).

When to Use Unicode-Mode Applications

Consider working with Unicode-mode applications only if you have any of the following situations:

- You need to enable users with different languages to view, in their own languages and character sets, information from a common database. For example, using alias tables in Japanese and German, users in Japan and Germany can view, in their own languages, information about a common product set.
- You need to handle object names longer than non-Unicode-mode applications support. For example, application and database names need to include more than eight characters, or you are working with a multi-byte character set and you need to handle more characters in object names. (For character restrictions, see Appendix A, “Limits.”)
- For a translated, multi-byte Analytic Services implementation, you have experienced what is called the “round-trip” problem. The round-trip problem can occur in communications between multi-byte operating systems and application programs where two different bit values can map to the same character. As Java applications, Administration Services and Deployment

Services always work in Unicode. No encoding conversions occur when these clients work with Unicode-mode applications and UTF-8-encoded text files; hence no round-trip conversion errors occur.

When deciding on using Unicode-mode applications, you should also consider the following points:

- Using non-Unicode text files with Unicode-mode applications requires an understanding of locales and care in managing locales. To prevent errors that can cause database corruption, using UTF-8-encoded files is recommended. For details, see "Managing File Encoding" on page 899.
- To work with Unicode-mode applications, custom client programs that were written to support non-Unicode-mode applications must be built to use the longer string lengths used by Unicode-mode applications. This task may be a simple re-build or may involve re-programming, depending on the design of the applications. Also, depending on how modified programs are coded, more memory may be required. For details, see the *API Reference*.

Chapter 40

Administering Unicode-Mode Applications

In most cases, administering Unicode-mode applications is no different than administering non-Unicode-mode applications. Working in multiple character sets in Unicode mode can require additional attention in the following areas:

Setting Up a Computer for Unicode Support

Consider installing the following tools (not provided by Hyperion Solutions) for working with UTF-8 encoded text on computers that manage Unicode-mode applications:

- To enable viewing of UTF-8 encoded text that contains non-ASCII characters, install a font that supports UTF-8 encoding.
- To support manual editing of data sources or other text files, install a Unicode editor that enables viewing, creating, and modifying UTF-8 encoded files and includes the UTF-8 signature. A Unicode editor is not necessary. Essbase Analytic Services provides the Analytic Services Unicode File Utility, which converts text files to UTF-8 encoding.

Several software manufacturers provide UTF-8 fonts and Unicode editors.

Defining Password Security

Analytic Services security is defined at the Analytic Server level. To create passwords, use characters encoded according to the code page specified in the ESSLANG value on Analytic Server. In a multiple character set environment, consider using the characters in the standard ASCII character set, in the range from 0 through 127. Standard ASCII includes the upper and lower case letters in the standard English alphabet and the numerals 0 through 9. These characters are common to all code pages as well as to UTF-8.

Naming Applications and Databases

When you name applications and databases, be sure to use characters supported by the locale of the computer.

Managing Unicode-Mode Applications

Working with Unicode-mode applications involves the following processes:

- "Setting Analytic Server to Unicode Mode" on page 895
- "Viewing the Unicode-Related Mode of Analytic Server" on page 895
- "Creating a New Unicode-Mode Application" on page 896
- "Migrating Applications to Unicode Mode" on page 896
- "Viewing the Unicode-Related Mode of an Application" on page 897

Setting Analytic Server to Unicode Mode

By setting Analytic Server to Unicode mode, you grant permission for creating Unicode-mode applications and for migrating applications to Unicode mode.

➤ To set Analytic Server to Unicode mode, use either of the following methods:

Tool	Topic	Location
Administration Services	Managing the Analytic Server Permission to Create Unicode-Mode Applications	*Essbase Administration Services Online Help*
MaxL	**alter system**	*Technical Reference*

Note: You can work with Unicode-mode applications without Analytic Server being set to Unicode mode.

Viewing the Unicode-Related Mode of Analytic Server

The Unicode-related mode of Analytic Server indicates whether or not Analytic Server has permission to create Unicode-mode applications and to migrate applications to Unicode mode. In Essbase Administration Services, the Unicode-related mode of Analytic Server is shown as a server property. In MaxL, it is displayed as the configuration, with the following values:

- 1, for non-Unicode-mode applications
- 2, for Unicode-mode applications

➤ To check to see whether Analytic Server is set to Unicode mode, use either of the following methods:

Tool	Topic	Location
Administration Services	Managing the Analytic Server Permission to Create Unicode-Mode Applications	*Essbase Administration Services Online Help*
MaxL	**display system**	*Technical Reference*

Creating a New Unicode-Mode Application

At the time that you create an application you can specify the application to be in Unicode mode.

➤ To create a new application and specify it to be in Unicode mode, use either of the following methods:

Tool	Topic	Location
Administration Services	Creating Unicode-Mode Applications	*Essbase Administration Services Online Help*
MaxL	**create application**	*Technical Reference*

Migrating Applications to Unicode Mode

When Analytic Server migrates an application to Unicode mode, the character encoding in the application files is converted to UTF-8 encoding. Text files such as calculation scripts, report scripts, and data sources are not converted to UTF-8 encoding. For Unicode-mode applications, text files such as calculation scripts, report scripts, and data sources can be encoded in UTF-8 or in non-Unicode locales. You can use the Analytic Services Unicode File Utility to convert non-Unicode-encoded files to UTF-8. For details about this utility, see the *Technical Reference*.

Note: If you choose to work with non-Unicode text files, make sure that you understand how Analytic Services determines the locales of non-Unicode text files. See "Managing File Encoding" on page 899.

Before migrating application files to Unicode mode, perform the following tasks:

- As with any major conversion, be sure to back up all application and database files in the application.
- If needed, apply the outline change files and then remove them. If present, outline change files reside in each database directory, with the database name as the file name and the extension `.chg`; for example, `\sample\basic\basic.chg`.

- To avoid mixing Unicode encoding and non-Unicode encoding in the logs, clear or remove log files after backing them up. Log files end with any of the following extensions: `.log`,`.xcp`, and `.olg`.
- Grant Analytic Server permission to migrate applications to Unicode mode. See "Setting Analytic Server to Unicode Mode" on page 895.

➤ To migrate an application from non-Unicode mode to Unicode mode, use either of the following methods:

Tool	Topic	Location
Administration Services	Migrating Applications to Unicode Mode	*Essbase Administration Services Online Help*
MaxL	**alter application**	*Technical Reference*

Viewing the Unicode-Related Mode of an Application

In Administration Services, the Unicode-related mode of an application is shown as an application property. In MaxL, it is displayed as the application type, with the following values:

- 2, for non-Unicode-mode applications
- 3, for Unicode-mode applications

➤ To see whether an application is a Unicode-mode application, use either of the following methods:

Tool	Topic	Location
Administration Services	Monitoring Applications	*Essbase Administration Services Online Help*
MaxL	**display application**	*Technical Reference*

Using Control Characters in Report Scripts

When working with reports for non-Unicode-mode applications, Report Writer uses language-specific codes that enable columnar data to line up precisely. Unicode does not contain language-specific formatting codes within its encoding. As a result, report columns may not line up precisely.

Non-Unicode-mode applications that use single-byte code pages continue to support the following Hexadecimal control characters in report scripts: `x20`, `x09`, `x0A`, `x0D`, `xB3`, `xC3`, `xC4`, `xC2`, `xC0`, `x0C`, `xB1`.

Unicode-mode applications and non-Unicode-mode applications that use multi-byte code pages support the following Hexadecimal control characters in report scripts: `x20`, `x09`, `x0A`, `x0C`, `x0D`.

Working with Partitions

Consider the following situations when designing and working with partitions:

- Partitions cannot connect non-Unicode-mode databases to Unicode-mode databases.
- All databases in a partition must be in the same encoding.
- Partitions across Unicode-mode databases can be administered by Unicode-mode clients; for example, Administration Services or MaxL scripts executed in Essbase Administration Services Console.
- Non-Unicode-mode outlines containing multi-byte characters cannot be synchronized by Unicode-mode clients such as Administration Services and MaxL scripts. However, you can use a client that is not Unicode-enabled, such as Application Manager, or a non-Unicode mode client, such as Essbase MaxL Shell (`essmsh`).

Working With Logs

By default, the agent log file is encoded to the locale specified by the ESSLANG variable defined for Analytic Server. You can use the configuration setting UNICODEAGENTLOG to tell Analytic Server to write the agent log in UTF-8 encoding. In UTF-8 encoding, with a font that supports it, all characters should be readable. For details about the UNICODEAGENTLOG configuration setting, see the *Technical Reference*.

Application and outline change log files for databases in Unicode-mode applications are UTF-8 encoded. You can use Log Viewer or Log Analyzer in Essbase Administration Services or a UTF-8 editor to view these application logs. You need a UTF-8 editor or viewer to view outline change logs. You cannot change the encoding of logs related to Unicode-mode applications to non-Unicode encoding.

Managing File Encoding

Analytic Services supports many non-Unicode encodings, as listed under "Supported Locales" on page 904. In addition, UTF-8 encoding is supported for Unicode-mode applications.

Because a bit combination can map to different characters in different encodings, you need to understand how Analytic Services works with file encodings—particularly if you intend to use non-Unicode-encoded files with Unicode-mode applications.

Note: For Unicode-mode applications shared across multiple locales, using UTF-8-encoded files is recommended. Using UTF-8 encoding is simpler because you do not have to keep track of locales and encoding. Also, using UTF-8 can be more efficient because Administration Server uses Unicode encoding internally and no internal conversion between formats is needed.

The following topics describe how Analytic Services determines the encoding of files and how you manage files with different encoding:

How the Encoding of a File Is Determined

With non-Unicode-mode applications, Analytic Services and Administration Services assume that character text is encoded to the locale specified by the ESSLANG value defined for Analytic Server.

When Analytic Services works with Unicode-mode applications, it encodes character text in UTF-8 encoding internally and stores character text in UTF-8. Export files are also encoded in UTF-8. When Analytic Services works with Unicode-mode applications, it can also handle non-Unicode-encoded input files such as script files, rules files, and data sources, converting them internally to UTF-8.

Note: For Unicode-mode applications, Analytic Services requires the query log settings file *dbname*`.cfg` to be encoded in UTF-8.

Analytic Services and Administration Services use file encoding indicators to know whether a file is encoded in UTF-8 or in one of the supported non-Unicode encodings. Encoding indicators are UTF-8 signatures or locale indicators. A locale indicator is optional for a non-Unicode input file whose encoding matches the locale of the Analytic Server. However, a locale indicator must be provided for a non-Unicode input file that is encoded different from the encoding specified in the locale of Analytic Server. For details about encoding indicators, see "Encoding Indicators" on page 901.

UTF-8-encoded text files must include the UTF-8 signature.

Administration Services uses the following process to determine the encoding of a non-Unicode-encoded file:

1. If a locale indicator is present in the file, Administration Services uses the specified encoding.
2. If no locale indicator is present and the file is located within an application, Administration Services uses the encoding specified in the locale of Analytic Server.
3. It no locale indicator is present and the file is not located within an application, Administration Services determines the encoding based on the type of file:
   - With text files, Administration Services prompts for the encoding.
   - With outline files and rules files, Administration Services uses the encoding specified in the locale of Analytic Server.

When Analytic Services performs a dimension build or data load, the rules file and data file can have different encodings; for example, the text in a rules file can be in UTF-8 encoding, and the data source can be encoded to a non-Unicode computer locale.

Note: When you use Administration Services Console to create script files or data sources, the appropriate encoding indicator is automatically included in the file. When you use any tool other than Administration Services Console to create text Unicode-encoded files, you must ensure that the UTF-8 signature is included. Non-Unicode-encoded text files require a locale indicator if the encoding is different than the locale of Analytic Server.

The following text Analytic Services system files are encoded to the locale specified by the ESSLANG value defined for Analytic Server.

- The configuration file (`essbase.cfg`)
- ESSCMD scripts

Encoding Indicators

To properly interpret text such as member names, Analytic Services must know how it is encoded. Many files contain an encoding indicator, but you may occasionally be prompted to specify the encoding of a file; for example, when Administration Services creates a file and stores it in a different location than the Analytic Server, or when Administration Services reads a file created by a non-Unicode release of Analytic Services. The type of encoding indicator depends on the type of file:

- Files that are internal to applications and databases and that users cannot directly edit are primarily binary files and do not contain any type of encoding indicator. Character text in these files is encoded to the application, which is either a Unicode-mode or non-Unicode mode application.
  - Text in Unicode-mode application files is UTF-8 encoded.
  - Text in non-Unicode-mode application files is encoded to the locale specified in the ESSLANG of the Analytic Server where the application was created.
- Binary files that you can edit include outline files and rules files. As needed, Analytic Services keeps track internally in outline files and rules files whether or not the character text is in UTF-8 encoding. If not UTF-8, Analytic Services uses an internal locale indicator to identify the locale used for character text encoding.

- The following text files that you can edit use a UTF-8 signature or a locale header to indicate their encoding:
  - Calculation scripts
  - Report scripts
  - MaxL scripts
  - Data sources for dimension builds and data loads
  - Alias table import files

    Note: Essbase Administration Services requires alias table import files to be UTF-8 encoded.

The *UTF-8 signature* is a mark at the beginning of a text file. The UTF-8 signature, visible in some third-party editors, indicates that the file is encoded in UTF-8. Many UTF-8 text editors can create the UTF-8 signature. You can also use the Analytic Services Unicode File Utility (ESSUTF8) to insert the UTF-8 signature into a file. For more information, see "Analytic Services Unicode File Utility" on page 906. When you create a file using Administration Services Console, a UTF-8 signature is automatically inserted in the file.

UTF-8-encoded text files must contain the UTF-8 signature.

The *locale header record* is an additional text record that identifies the encoding of the non-Unicode-encoded text file. You can add the locale header at the time you create the file or you can use the Analytic Services Unicode File Utility to insert the locale header. For details about the locale header, see "Locale Header Records" on page 903.

Note: Do not combine a UTF-8 signature and locale header in the same file. If a text file contains both types of encoding indicators, the file is interpreted as UTF-8 encoded, and the locale header is read as the first data record.

CAUTION: Do not use non-Unicode-encoded files containing locale indicators with Analytic Server installations that are not Unicode enabled, that is, installations that used a release prior to Release 7.0. The Analytic Services Unicode File Utility (ESSUTF8) enables you to remove locale indicators.

Locale Header Records

The locale header record is an additional text record that identifies the encoding of the non-Unicode text file. You can insert the locale header record as the first record in a non-Unicode-encoded text file when you create the file or you can use the Analytic Services Unicode File Utility (ESSUTF8) program to add the header. The locale header record has the following format:

```
//ESS_LOCALE <locale-name>
```

<*locale-name*> is a supported Global C locale in the format that is used for the ESSLANG variable:

```
<language>_<territory>.<code page name>@<sortsequence>
```

Note: Analytic Services consults only the <*code page name*> portion of the record. The <*sortsequence*> specification does not affect sort sequences in report scripts.

See "Supported Locales" on page 904 for a list of supported Global C locales.

The following example displays a locale header for a specific Russian code page:

```
//ESS_LOCALE Russian_Russia.ISO-8859-5@Default
```

The following rules also apply:

- After <*locale-name*>, use a blank, tab or <end of line> to end the header.
- You can include blanks or tabs in the following locations:
  - Before the keyword "`//ESS_LOCALE`"
  - Between "`//ESS_LOCALE`" and <*locale-name*>
  - After the locale specification

CAUTION: Do not insert a locale header record in a UTF-8 encoded file. The file is interpreted as UTF-8 encoded, and the locale header is read as the first data record.

For compatibility, when Administration Services Console saves calculation scripts on Analytic Server installations that are not Unicode-enabled, that is, installations that use a release prior to Release 7.0, it uses a different format. Instead of prefixing the locale with `//`, Administration Services Console inserts the locale header between the calculation script comment indicators, `/* */`.

Supported Locales

The following supported locales can be used as locales in locale header records and in the Analytic Services Unicode File Utility (ESSUTF8) and as ESSLANG values.

```
Arabic_SaudiArabia.ISO-8859-6@Default
Arabic_SaudiArabia.MS1256@Default
Croatian_Croatia.ISO-8859-2@Croatian
Croatian_Croatia.MS1250@Croatian
CyrillicSerbian_Yugoslavia.ISO-8859-5@Default
CyrillicSerbian_Yugoslavia.MS1251@Default
Czech_CzechRepublic.ISO-8859-2@Czech
Czech_CzechRepublic.MS1250@Czech
Danish_Denmark. ISO-8859-15@Danish
Danish_Denmark.IBM500@Danish
Danish_Denmark.Latin1@Danish
Danish_Denmark.MS1252@Danish
Dutch_Netherlands.IBM037@Default
Dutch_Netherlands.IBM500@Default
Dutch_Netherlands.ISO-8859-15@Default
Dutch_Netherlands.Latin1@Default
Dutch_Netherlands.MS1252@Default
English_UnitedStates.IBM037@Binary
English_UnitedStates.IBM285@Binary
English_UnitedStates.IBM500@Binary
English_UnitedStates.Latin1@Binary
English_UnitedStates.MS1252@Binary
English_UnitedStates.US-ASCII@Binary
Finnish_Finland.IBM500@Finnish
Finnish_Finland.ISO-8859-15@Finnish
Finnish_Finland.Latin1@Finnish
Finnish_Finland.MS1252@Finnish
French_France.IBM297@Default
French_France.IBM500@Default
French_France.ISO-8859-15@Default
French_France.Latin1@Default
French_France.MS1252@Default
German_Germany.IBM273@Default
German_Germany.IBM500@Default
German_Germany.ISO-8859-15@Default
German_Germany.Latin1@Default
German_Germany.MS1252@Default
Greek_Greece.ISO-8859-7@Default
Greek_Greece.MS1253@Default
Hebrew_Israel.ISO-8859-8@Default
Hebrew_Israel.MS1255@Default
```

```
Hungarian_Hungary.ISO-8859-2@Hungarian
Hungarian_Hungary.MS1250@Hungarian
Italian_Italy.IBM280@Default
Italian_Italy.IBM500@Default
Italian_Italy.ISO-8859-15@Default
Italian_Italy.Latin1@Default
Italian_Italy.MS1252@Default
Japanese_Japan.IBM930@Binary
Japanese_Japan.JapanEUC@Binary
Japanese_Japan.MS932@Binary
Korean_Korea.MS1361@Binary
Korean_Korea.MS949@Binary
Norwegian_Norway.IBM500@Danish
Norwegian_Norway.ISO-8859-10@Danish
Norwegian_Norway.ISO-8859-15@Danish
Norwegian_Norway.ISO-8859-4@Danish
Norwegian_Norway.Latin1@Danish
Norwegian_Norway.MS1252@Danish
Polish_Poland.ISO-8859-2@Polish
Polish_Poland.MS1250@Polish
Portuguese_Portugal.IBM037@Default
Portuguese_Portugal.IBM500@Default
Portuguese_Portugal.ISO-8859-15@Default
Portuguese_Portugal.Latin1@Default
Portuguese_Portugal.MS1252@Default
Romanian_Romania.ISO-8859-2@Romanian
Romanian_Romania.MS1250@Romanian
Russian_Russia.ISO-8859-5@Default
Russian_Russia.MS1251@Default
Serbian_Yugoslavia.ISO-8859-2@Default
Serbian_Yugoslavia.MS1250@Default
SimplifiedChinese_China.IBM935@Binary
SimplifiedChinese_China.MS936@Binary
SimplifiedChinese_China.UTF-8@Binary
Slovak_Slovakia.ISO-8859-2@Slovak
Slovak_Slovakia.MS1250@Slovak
Slovenian_Slovenia.ISO-8859-10@Slovenian
Slovenian_Slovenia.ISO-8859-2@Slovenian
Slovenian_Slovenia.ISO-8859-4@Slovenian
Slovenian_Slovenia.MS1250@Slovenian
Spanish_Spain.IBM500@Spanish
Spanish_Spain.ISO-8859-15@Spanish
Spanish_Spain.Latin1@Spanish
Spanish_Spain.MS1252@Spanish
Swedish_Sweden.IBM500@Swedish
Swedish_Sweden.ISO-8859-15@Swedish
```

```
Swedish_Sweden.Latin1@Swedish
Swedish_Sweden.MS1252@Swedish
Thai_Thailand.MS874@Thai
TraditionalChinese_Taiwan.EUC-TW@Binary
TraditionalChinese_Taiwan.IBM937@Binary
TraditionalChinese_Taiwan.MS950@Binary
Turkish_Turkey.ISO-8859-3@Turkish
Turkish_Turkey.ISO-8859-9@Turkish
Turkish_Turkey.MS1254@Turkish
Ukrainian_Ukraine.ISO-8859-5@Ukrainian
Ukrainian_Ukraine.MS1251@Ukrainian
```

Analytic Services Unicode File Utility

You can use the Analytic Services Unicode File Utility to convert non-Unicode-encoded text files to UTF-8 encoding or to add or delete encoding indicators. This program supports the following operations:

- Converting non-Unicode-encoded text files to UTF-8 encoding, including the UTF-8 signature
- Adding UTF-8 signatures to UTF-8-encoded text files
- Inserting a locale header record at the beginning of non-Unicode-encoded text files
- Adding a locale indicator to outline files (`.otl`) or rules files (`.rul`)
- Removing locale indicators from files. (The utility cannot remove UTF-8 signatures.)

Located in the *ARBORPATH*`\bin` directory, the Analytic Services Unicode File Utility is called `essutf8.exe` (in Windows) or `ESSUTF8` (in UNIX). You can use the utility program with the following files:

- Calculation scripts
- Report scripts
- MaxL scripts
- Text data sources for dimension builds and data loads
- Alias table import files
- Outline files
- Rules files

For information about this utility and its command syntax, see the *Technical Reference*.

Part VIII

Maintaining Analytic Services

This part describes how to manage data storage for Analytic Services databases and how to ensure data integrity:

- Chapter 41, "Running Analytic Servers, Applications, and Databases," describes the Analytic Services Agent and describes the different ways to start and stop Analytic Server, applications, and databases.
- Chapter 42, "Managing Applications and Databases," introduces you to the kinds of files that Analytic Services uses, describes operations you can perform on Analytic Services applications and databases, and discusses some of the issues you face when working with Analytic Services files on different platforms.
- Chapter 43, "Monitoring Data, Applications, and Databases," describes each Analytic Services log, including its contents, and all the actions you can perform on each log.
- Chapter 44, "Managing Database Settings," introduces you to the Analytic Services Kernel and describes how the kernel stores and accesses data, and how you can adjust the database settings to control that access and storage.
- Chapter 45, "Allocating Storage and Compressing Data," describes how Analytic Services stores data on disk, explains how to use disk volumes, and describes the compression methods Analytic Services uses to compress data.
- Chapter 46, "Ensuring Data Integrity," explains how Analytic Services handles transactions and locking and discusses other ways that Analytic Services protects data. This chapter describes transactions, isolation levels, and data recovery, and tells you how to configure Analytic Services to ensure the integrity of your data.
- Chapter 47, "Backing Up and Restoring Data," describes how to back up databases and how to restore data from backups.

Chapter 41

Running Analytic Servers, Applications, and Databases

This chapter describes the various methods used to run Essbase Analytic Server and its associated applications and databases. It defines the role of the Agent, which is the central organizing subsystem for Analytic Server, and describes different methods of starting and stopping Analytic Servers, applications, and databases.

This chapter includes the following sections:

Understanding Analytic Services Executable Files

Seven main executable files contain the server and client software for Analytic Services:

Table 35: Main Analytic Services Executable Files

Executable File	Description	More Information
`essbase.exe`	Analytic Server Agent process	"Understanding the Agent" on page 911
`esssvr.exe`	Application process	"Starting and Stopping Applications" on page 930
`essmsh.exe`	MaxL Shell	*Technical Reference*
`esscmd.exe`	ESSCMD command-line client interface	*Technical Reference*
`startEAS.exe`	Administration Server executable	*Essbase Administration Services Online Help*
`admincon.exe`	Administration Services Console application	*Essbase Administration Services Online Help*

The files that start with `ess` are stored in the `\`*`ARBORPATH`*`\bin` directory (`/`*`ARBORPATH`*`/bin` on UNIX). `adminsvr.exe` is stored in *`ARBORPATH`*`\eas\server\bin` directory (*`ARBORPATH`*`/eas/bin` on UNIX), and `admincon.exe` is stored in *`ARBORPATH`*`\eas\console\bin` directory (*`ARBORPATH`*`/eas/bin` on UNIX). On UNIX systems, these files do not have the `.exe` extension.

You can launch any executable file by typing its name at the operating system command prompt. On Windows, you can also launch a program by double-clicking the file in Explorer, double-clicking a desktop icon, or entering its name in the dialog box displayed when you select Run from the Start menu.

Understanding the Agent

The Agent Process is the Analytic Server process that starts and stops all applications and acts as the traffic coordinator for the Analytic Server. Use this section to understand more about the Agent:

- “Flow of Communications Events” on page 912
- “Multithreading” on page 913
- “Displaying of a List of Agent Commands” on page 914

The agent log is called the Analytic Server log. For information on viewing the Analytic Server log, see “Viewing the Analytic Server and Application Logs” on page 997.

When you start Analytic Server in the foreground, the Agent becomes active in an operating system window. You should see information like the following appear in the operating system window:

Figure 218: Sample Agent Output to Operating System Window

```
Unlimited login system
Hyperion Essbase Analytic Server - X.X
Copyright 1991-2002 Hyperion Solutions Corporation.
US Patent Number 5,359,724
All Rights Reserved.
Serial number:  XXXXXXXXXXX-XXXXXXXXXX
Registered to:  admin
                hyperion
Please type the system password: ********
Startup sequence completed
Security is enabled
Logins are enabled
Essbase Default Storage type is Multidimensional
[Wed Dec 06 14:00:20 2002]Local/ESSBASE0///Info(1051051)
Hyperion Essbase Analytic Server - started
Waiting for Client Requests...
```

In the Agent, you can enter administrative commands and monitor the behavior of Analytic Services. The Agent is accessible only from the server console on the Analytic Server computer. The server console is the primary terminal, or monitor, connected to the server computer. The server computer is the computer on which Analytic Server runs.

The Agent executable file, `essbase.exe` (`ESSBASE` on UNIX systems), is stored in the `\ARBORPATH\bin` directory. Launching this executable file starts Analytic Server. You can start Analytic Server in the foreground or as a background process. If you start Analytic Server in the background, the terminal becomes free for other input, and the Agent activities are not visible in the terminal. For a detailed explanation, see "Starting Analytic Server as a Background Process" on page 918.

Flow of Communications Events

When a user logs on to Analytic Server, these events occur:

1. An Analytic Services client logs on, using a predefined address:
   - For TCP/IP connections, the address is port number 1423 unless it has been changed using the configuration setting AGENTPORT.
   - For Named Pipes connections, the address is `\\pipe\essbase0`.
2. The user selects an application/database combination.
3. The Agent compares the requested application with applications currently running. If the specified application is already running, the Agent does not need to do anything. If the requested application is not already running, the Agent initiates startup, and the following events occur:
   a. The Agent assigns a dynamic port for the application server or creates a dynamic name for a named pipe.
   b. The application server returns the port number for the application to the Agent and to the client. Port number information is stored at run time in Analytic Services API.
4. The Agent sets the application server as active with the current user and security information. When the client later sends a data request to the application server, the security information captured by the API is embedded in the request.
5. The Agent sends the client query to the application server (ESSSVR).

6. For every request that the client makes, the application server causes the following events to occur:
   a. Connect
   b. Send request
   c. Get response
   d. Send data
   e. Receive data
   f. Disconnect

Multithreading

`essbase.exe` and `esssvr.exe` (ESSBASE and ESSSVR on UNIX) are multithreaded applications. By default, the number of threads is based on the number of licensed ports that are purchased, as shown in Table 36. The number of ports represents the number of concurrent connections that Analytic Services supports. Analytic Services provides one reserve port for the system administrator. The system administrator uses the reserve port to log off one or more users when all other ports are in use.

Table 36: Licensed Ports and Multithreading

Number of Licensed Ports	Default Number of Threads
1–5 ports	5
6–10 ports	10
11 or more ports	20

You can set the number of threads for the Agent or Analytic Server in the `essbase.cfg` file. For complete details on setting the number of threads for the Agent and the server, refer to the AGENTTHREADS, AGTSVRCONNECTIONS, and SERVERTHREADS configuration settings in the *Technical Reference*.

Displaying of a List of Agent Commands

➤ To display a list of all available Agent commands, press Enter in the operating system window where you started Analytic Server in the foreground. For an example of Agent output, see Figure 218 on page 911.

Use the following table to understand the function of each Agent command displayed, and the MaxL, ESSCMD, or Administration Services equivalents:

Table 37: Agent Commands and MaxL, ESSCMD, or Administration Services Equivalents

Agent Command	Function	MaxL, ESSCMD, or Administration Services Equivalent
START *appname*	Starts the specified application.	• MaxL: **alter system load application** *appname*; • ESSCMD: LOADAPP • Administration Services: Right-click menu off application node in Enterprise View
STOP *appname*	Stops the specified application.	• MaxL: **alter system unload application** *appname*; • ESSCMD: UNLOADAPP • Administration Services: Right-click menu off application node in Enterprise View
USERS	Displays a list of all users connected to the Analytic Server. The following information is displayed: • Names of all users connected to the Analytic Server • Total number of ports installed • Total number of existing connections • Application to which each user is connected • Database to which each user is connected	• MaxL: **display user;** (lists all users and shows which users are logged on) • ESSCMD: LISTUSERS (lists all users) • Administration Services: N/A

Table 37: Agent Commands and MaxL, ESSCMD, or Administration Services Equivalents (Continued)

Agent Command	Function	MaxL, ESSCMD, or Administration Services Equivalent
PORTS	Displays the number of ports installed on the Analytic Server and the number of ports in use.	• MaxL: **display system;** (to display available unused ports) • ESSCMD: N/A • Administration Services: Analytic Server Properties window, Statistics tab
LOGOUTUSER *user*	Disconnects a user from the Analytic Server and frees a port. This command requires the Analytic Services system password.	• MaxL: **alter system logout session by user** *username*; • ESSCMD: LOGOUTUSER • Administration Services: Sessions window
PASSWORD	Changes the system password that is required to start the Analytic Server. This command requires the Analytic Services system password.	• MaxL: **alter user** *system supervisor* **set password** *password*; • ESSCMD: SETPASSWORD • Administration Services: N/A
COMPACT	Enables defragmentation of the security file when the Agent is running. **Note:** Analytic Services compacts the security file automatically each time the Agent is stopped.	• MaxL **alter system compact security file;** • ESSCMD: N/A • Administration Services: N/A
DUMP *filename*	Dumps information from the Analytic Services security system to a specified file in ASCII format. If you do not supply a path with the file name, the file is saved to the bin directory, for example, `\essbase\bin`. This command requires the Analytic Services system password.	N/A

Table 37: Agent Commands and MaxL, ESSCMD, or Administration Services Equivalents (Continued)

Agent Command	Function	MaxL, ESSCMD, or Administration Services Equivalent
VERSION	Displays the Analytic Server software version number.	• MaxL: **display system;** • ESSCMD: GETVERSION • Administration Services: Analytic Server Properties window, License tab
HELP	Lists all valid Agent commands and their respective functions. Same as pressing Enter.	N/A
QUIT and EXIT	Shuts down all open applications and stops Analytic Server.	• MaxL: **alter system shutdown;** • ESSCMD: SHUTDOWNSERVER • Administration Services: Right-click menu off server node in Enterprise View

Starting and Stopping

Use these sections for instructions about starting and stopping Analytic Servers, applications, and databases:

- "Starting and Stopping Analytic Server" on page 917
- "Starting Analytic Server Remotely from Administration Services Console" on page 920
- "Starting and Stopping Administration Services" on page 930
- "Starting and Stopping Applications" on page 930
- "Starting and Stopping Databases" on page 935

Starting and Stopping Analytic Server

You can start Analytic Server in a variety of ways. You need Supervisor permissions to start Analytic Server.

Use these sections for instructions about starting and stopping Analytic Server:

Note: For instructions about starting Analytic Server from Administration Services, see "Starting Analytic Server Remotely from Administration Services Console" on page 920.

Starting Analytic Server

Analytic Server can be started in the foreground using several methods.

➤ To start Analytic Server in the foreground, use any of the following methods:

Operating System	Instructions
UNIX	Enter the following command at the operating system prompt. ESSBASE *password*
Windows	Use any of the following methods: • Enter the following command at the operating system prompt. ESSBASE *password* • Select Analytic Server from the Hyperion Solutions program group on the Start menu. • Double-click `essbase.exe`, located in the `bin` directory, and enter the password when prompted. • From the Start menu, select Run, and type ESSBASE *password* in the Open field. *password* is the password to access Analytic Server.

Any of these actions start the Agent in the operating system command console. In the Agent, you can enter commands or monitor Analytic Services activities (see "Understanding the Agent" on page 911).

You cannot start from ESSCMD or MaxL. To start Analytic Server from Administration Services Console, see "Starting Analytic Server Remotely from Administration Services Console" on page 920.

Starting Analytic Server as a Background Process

Analytic Server can run as a background process on the Analytic Server computer.

System administrators might choose to run Analytic Server as a background process when working in batch mode. Running in batch mode means, for example, that the administrator can start Analytic Server, launch MaxL or ESSCMD, log on to Analytic Server, load data, run calculation scripts or reports, stop Analytic Server, and perform a backup, all from a single UNIX shell script or a Windows NT `.bat` file.

System administrators running Analytic Server on Windows NT or Windows 2000 might also choose to run Analytic Server as a background process in order to use Windows settings to give a performance boost to applications running in the foreground.

If you start Analytic Server in the background, these conditions apply:

- You do not have access to Agent commands.
- You cannot shut down Analytic Server from the Agent. You must use MaxL or ESSCMD to shut down Analytic Server.
- You cannot access the application server window to monitor a running application. You must access this information from the application log (*ARBORPATH*`/app/`*appname*`/`*appname*`.log`).
- You cannot monitor Analytic Server activity using the Agent. You must access this information from the Analytic Server log (*ARBORPATH*`/essbase.log`).

➤ To start Analytic Server in the background on UNIX, or on Windows systems utilizing a UNIX-like shell such as MKS, enter the following command at the operating system command prompt:

```
essbase password -b &
```

You can start Analytic Server without using the ampersand (&) at the end of the command, but if you do not use "&," the command prompt is not returned after the Analytic Server is started.

Note: The ampersand (&) used in the above context has no effect when used on Windows, unless you are using a UNIX-like shell such as MKS. Analytic Server starts in the background, but control of the command prompt is not returned. You may need to press the Enter key twice in Windows before the command prompt returns.

➤ On UNIX systems, to find out if Analytic Server is already running in the background, type this command at the operating system command prompt:

```
ps -ef | grep ESS
```

If the Analytic Server is running in the background, it appears in the process list:

```
essbase password -b &
```

For information about hiding the password from the UNIX process list, see the *Essbase Analytic Services Installation Guide*.

For information about how to run Analytic Server as a Windows NT Service, or to see if Analytic Server is already installed as a Windows NT Service, see the *Essbase Analytic Services Installation Guide*.

Changing the System Password

Using an Agent command, you can change the password that is required to start the Analytic Server.

Note: Changing the system password does not change the connection password that is established for the Analytic Services system supervisor.

➤ To change the Analytic Server system password, perform the following tasks:

1. Type **password** at the command prompt of the Agent in the server console.
2. Type the old (current) system password.
3. Type the new system password.
4. Retype the new system password.

Analytic Services verifies that the system password has been updated.

For instructions on how to change passwords for users or groups, see "Editing Users" on page 845.

Stopping Analytic Server

You need Supervisor permissions to stop Analytic Server.

➤ To stop (also "shut down") Analytic Server and all running applications, use any of the following methods:

Tool	Topic	Location
Agent	quit exit	Enter the command at the command prompt for the Agent in the Analytic Server console window
Administration Services	Stopping Analytic Server	*Essbase Administration Services Online Help*
MaxL	**alter system shutdown**	*Technical Reference*
ESSCMD	SHUTDOWNSERVER	*Technical Reference*

If you stop the Agent by closing the server console window or by pressing Ctrl + C, the next time you start the database, Analytic Services rolls back any transactions that were in progress. See "Rollback with Committed Access" on page 1060 or "Rollback with Uncommitted Access" on page 1062 for a description of the roll back process (undoing all transactions after the last committed transaction).

Starting Analytic Server Remotely from Administration Services Console

You can start Analytic Server remotely from the Enterprise View tree in Administration Services Console. To enable this functionality, you configure and start Remote Start Server on the Analytic Server machine. Remote Start Server then handles requests from Administration Services to start Analytic Server.

Each Analytic Server installation includes all files necessary to configure and start Remote Start Server. If you want to start different Analytic Servers from Administration Services Console, you must configure and start a Remote Start Server instance for each Analytic Server.

To start Analytic Servers from previous releases, you must first download Remote Start Server from Hyperion Developer Network and install it on each Analytic Server machine.

Use the following topics for instructions about starting Analytic Server remotely:

- “Configuring Remote Start Server” on page 921
- “Starting and Stopping Remote Start Server” on page 928
- “Running Remote Start Server as a Windows Service” on page 929
- “Starting Analytic Server from Enterprise View” on page 929

Configuring Remote Start Server

Before starting Remote Start Server, you need to configure it if any of the following conditions apply:

- Analytic Server is configured to run on a non-default port.
- Environment variables need to be set when Analytic Server is started.
- The default port used for Remote Start Server (9010) is being used by another program.
- More than one instance of Analytic Server is installed on a single machine.

If none of the conditions apply, you are ready to start Remote Start Server. Skip to “Starting and Stopping Remote Start Server” on page 928.

A configuration file for Remote Start Server is installed with Analytic Server. The following topics describe the configuration file and tell you how to modify the file for different purposes:

- “About the Configuration File” on page 922
- “Specifying a Non-default Analytic Server Port” on page 923
- “Setting Environment Variables for Startup” on page 924
- “Specifying a Non-default Remote Start Server Port” on page 925
- “Configuring Remote Start Server for Multiple Instances of Analytic Server” on page 926
- “Sample Configuration File” on page 928

About the Configuration File

Each Analytic Server installation contains a configuration file that you can use to configure Remote Start Server. The configuration file is:

ARBORPATH\bin\server.properties

The default configuration file contains the following lines:

```
Server1=localhost
localhostExecutable=PathToEssbase.exe
```

Where *PathToEssbase.exe* is the location of the Analytic Server executable file.

Keep in mind the following information when editing the configuration file:

- You can edit the file in any text editor.
- On Windows, you must always escape the backslash (\) with an additional backslash (\\). See the examples in the following sections.
- The structure of each line in the file is:

  ```
  key=keyValue
  ```

- The value for the `Server1` key may be any one of the following:
  - `localhost`
  - *machineName*
  - *machineName*:*port*
- If you change the value of `Server1` to `machineName`, you need to use `machineName` in place of `localhost` as the key prefix for any rows you add to the file.

  As an example on UNIX:

  ```
  Server1=jdoe2
  jdoe2Executable=/essbase/bin/essbase.exe
  ```

- If you change the value of `Server1` to `machineName:port`, you need to use `Server1` in place of `localhost` as the key prefix for any rows you add to the file. For example (on Windows):

  ```
  Server1=jdoe2:4050
  Server1Executable=c:\\hyperion\\essbase\\essbase.exe
  ```

- When adding environment rows to the file, you first specify how many environment rows you are adding. For example:

  ```
  localhostEnvRows=3
  localhostEnv1=Variable=VariableValue
  localhostEnv2=Variable=VariableValue
  localhostEnv3=Variable=VariableValue
  ```

  For examples, see "Setting Environment Variables for Startup" on page 924

Specifying a Non-default Analytic Server Port

If Analytic Server is configured to run on a non-default port, in the Remote Start Server configuration file, you must specify the port number being used for Analytic Server.

If you are using a non-default port because you are using more than one instance of Analytic Server on a single machine, see "Configuring Remote Start Server for Multiple Instances of Analytic Server" on page 926.

➤ To specify a non-default Analytic Server port:

1. From the Analytic Server machine, open the following file:

   ARBORPATH\bin\server.properties

2. Change the following lines, supplying the appropriate information for the italicized text:

   ```
   Server1=localhost
   localhostExecutable=PathToEssbase.exe
   ```

   to

   ```
   Server1=MachineName:PortNumber
   Server1Executable=PathToEssbase.exe
   ```

   As an example for Windows:

   ```
   Server1=jdoe2:4050
   Server1Executable=c:\\hyperion\\essbase\\bin\\essbase.exe
   ```

Setting Environment Variables for Startup

If environment variables need to be set when Analytic Server is started remotely, you need to add environment information to the Remote Start Server configuration file:

- On Windows, unless you are using more than one instance of Analytic Server on the same machine, you do not need to set environment variables. For information about configuring multiple instances of Analytic Server, see "Configuring Remote Start Server for Multiple Instances of Analytic Server" on page 926.
- On UNIX, the environment must be set properly when Analytic Server is started. If the environment is not set, in the Remote Start Server configuration file, you must specify environment variable information to be used for startup. Each UNIX platform has different environment variable requirements; for information about the environment variables that are required for each UNIX platform, see *Essbase Analytic Services Installation Guide*.

➤ To set environment variables for startup:

1. From the Analytic Server machine, open the following file:

   ARBORPATH`\bin\server.properties`

2. Add lines for each environment variable you need to specify, using the format described in "About the Configuration File" on page 922. The number of environment rows that need to be set depends on your platform.

The following sections show examples of how to set environment variables for different UNIX platforms, using the default Analytic Server port and an English ESSLANG value).

Solaris Example:

```
Server1=localhost
localhostEnvRows=4
localhostEnv1=ARBORPATH=/vol1/essbase
localhostEnv2=LD_LIBRARY_PATH=/vol1/common/ODBC/Merant/
    4.2/lib:/vol1/essbase/bin;$LD_LIBARY_PATH
localhostEnv3=ESS_JVM_OPTION1=-Xusealtsigs
localhostEnv4=ESSLANG=English_UnitedStates.Latin1@Binary
localhostExecutable=/vol1/essbase/bin/ESSBASE
```

AIX Example:

```
Server1=localhost
localhostEnvRows=3
localhostEnv1=ARBORPATH=/vol1/essbase
localhostEnv2=LIBPATH=/vol1/common/ODBC/Merant/
    4.2/lib:/vol1/essbase/bin;$LIBPATH
localhostEnv3=ESSLANG=English_UnitedStates.Latin1@Binary
localhostExecutable=/vol1/essbase/bin/ESSBASE
```

HP-UX Example

```
Server1=localhost
localhostEnvRows=3
localhostEnv1=ARBORPATH=/vol1/essbase
localhostEnv2=SHLIB_PATH=/vol1/common/ODBC/Merant/
    4.2/lib:/vol1/essbase/bin;$SHLIB_PATH
localhostEnv3=ESSLANG=English_UnitedStates.Latin1@Binary
localhostExecutable=/vol1/essbase/bin/ESSBASE
```

Specifying a Non-default Remote Start Server Port

By default, the Remote Start Server is configured to run on port 9010. If this port is being used by another program, you must specify a different port number before you start Remote Start Server. The port number must be specified in the Remote Start Server configuration file on the Analytic Server machine and in a configuration file on the Administration Server machine.

➤ To specify a non-default Remote Start Server port:

1. From the Analytic Server machine, open the following file:

   ARBORPATH\bin\server.properties

2. Add the following line, supplying the appropriate information for the italicized text:

   ServerPort=*PortNumber*

   For example:

   ServerPort=9030

3. From the Administration Server machine, open the following file:

   EASPATH\eas\server\olapadmin.properties

4. Add one of the following lines:

   - If you want Remote Start Server to use a specific port for a specific Analytic Server, add the following line, supplying the appropriate information for the italicized text:

     AnalyticServerName.REMOTE_START_SERVER=*PortNumber*

     For example:

     jdoe2.REMOTE_START_SERVER=9030

   - If you want Remote Start Server to use the same, non-default port for all Analytic Servers, add the following line, supplying the appropriate information for the italicized text:

     REMOTE_START_SERVER=*PortNumber*

     For example:

     REMOTE_START_SERVER=9030

Configuring Remote Start Server for Multiple Instances of Analytic Server

If more than one instance of Analytic Server is installed on one machine, you need to configure and start Remote Start Server for only one of the installations. Choose one of the Analytic Server installations, and add configuration information for both Analytic Server instances to the configuration file that is installed with that instance.

Note: For information about installing multiple Analytic Server agents, see "Installing Additional Instances of Analytic Server: Windows" on page 942 or "Installing Additional Instances of Analytic Server: UNIX" on page 945.

➤ To configure Remote Start Server for multiple instances of Analytic Server on one machine:

1. Select one of the Analytic Server installations, and open the following file from the directory structure under that installation:

   ARBORPATH\bin\server.properties

2. Add the following lines to the file, supplying the appropriate information for the italicized text:

   ```
   Server2=MachineName:PortNumber
   Server2EnvRows=1
   Server2Env1=ARBORPATH=ARBORPATHvalue
   Server2Executable=PathToEssbase.exe
   ```

   As an example on Windows:

   ```
   Server2=jdoe2:4050
   Server2EnvRows=1
   Server2Env1=ARBORPATH=c:\\hyperion\\essbase
   Server2Executable=c:\\hyperion\\essbase2\\
   bin\\essbase.exe
   ```

   Note that ARBORPATH must be set explicitly for the second server instance. Also, for UNIX platforms, if the environment is not already set for either Analytic Server instance, you need to specify environment variable information in the Remote Start Server configuration file. For information about the environment variables that are required for each UNIX platform, see *Essbase Analytic Services Installation Guide*.

Windows Example

The following example for Windows shows the default configuration for `localhost` and the additional configuration needed for a second instance of Analytic Server (`jdoe2`), which runs on port 4050.

```
Server1=localhost
localhostExecutable=c:\\hyperion\\essbase\\
    bin\\essbase.exe

Server2=jdoe2:4050
Server2EnvRows=1
Server2Env1=ARBORPATH=c:\\hyperion\\essbase2
Server2Executable=c:\\hyperion\\essbase2\\
    bin\\essbase.exe
```

UNIX Example (Solaris)

The following example for Solaris shows the configuration for `localhost` and for the second instance of Analytic Server (`jdoe2`), which runs on port 4050. In this example, the environment is set for both Analytic Server instances.

```
Server1=localhost
localhostEnvRows=4
localhostEnv1=ARBORPATH=/vol1/essbase
localhostEnv2=LD_LIBRARY_PATH=/vol1/common/ODBC/Merant/
    4.2/lib:/vol1/essbase/bin;$LD_LIBARY_PATH
localhostEnv3=ESS_JVM_OPTION1=-Xusealtsigs
localhostEnv4=ESSLANG=English_UnitedStates.Latin1@Binary
localhostExecutable=/vol1/essbase/bin/ESSBASE

Server2=jdoe2:4050
Server2EnvRows=4
Server2Env1=ARBORPATH=/vol2/essbase2
Server2Env2=LD_LIBRARY_PATH=/vol2/common/ODBC/Merant/
    4.2/lib:/vol2/essbase2/bin:$LD_LIBRARY_PATH
Server2Env3=ESS_JVM_OPTION1=-Xusealtsigs
Server2Env4=ESSLANG=English_UnitedStates.Latin1@Binary
Server2Executable=/vol2/essbase2/bin/ESSBASE
```

Sample Configuration File

The following sample configuration file for Windows sets the Analytic Server port to 4050; sets the ARBORPATH environment variable; and sets the Remote Start Server port to 9030:

```
Server1=jdoe2:4050
Server1EnvRows=1
Server1Env1=ARBORPATH=c:\\hyperion\\essbase
Server1Executable=c:\\hyperion\\essbase\\bin\\essbase.exe
ServerPort=9030
```

Starting and Stopping Remote Start Server

When you are sure that Remote Start Server is properly configured, you can start and stop it as needed. After Remote Start Server is started, you can start Analytic Server.

Note: On UNIX platforms, Remote Start Server can start an Analytic Server only if the Analytic Server is installed by the same user ID that was used to start Remote Start Server. If Remote Start Server is started by the root user, then Remote Start Server can start an Analytic Server that was installed by any user.

➤ To start Remote Start Server:

- On Windows platforms, launch

  ARBORPATH\bin\remoteStart.exe

- On UNIX platforms, launch

  ARBORPATH/bin/remotesvr

➤ To stop Remote Start Server:

- On Windows platforms, launch

  ARBORPATH\bin\stopsvr.exe

- On UNIX platforms, launch

  ARBORPATH/bin/stopsvr

Running Remote Start Server as a Windows Service

You can install and start Remote Start Server as a Windows service.

➤ To install and start Remote Start Server as a Windows Service, launch

ARBORPATH\bin\install_service.bat

This file installs Remote Start Server as a Windows service (named "Essbase Remote Start Server") and starts the service. You can then manage the service by using Windows Control Panel or by using the `net start` and `net stop` commands.

➤ To remove the "Essbase Remote Start Server" Windows Service, launch

ARBORPATH\bin\remove_service.bat

Starting Analytic Server from Enterprise View

After Remote Start Server is started, you can start Analytic Server from Enterprise View in Administration Services Console:

To start Analytic Server:

1. From Enterprise View, select the Analytic Server node that you want to start.
2. Right-click and select **Start** from the pop-up menu.

3. When prompted, enter the password for Analytic Server.

   The request is sent to Remote Start Server, which then starts Analytic Server. If Remote Start Server is not running, an error message is displayed in Administration Services Console.

Note: Analytic Server starts in the background on all platforms, with the password hidden.

Starting and Stopping Administration Services

Before you start Administration Services, make sure that the Analytic Servers you want to manage are started.

To start Administration Services, first start Administration Server, and then start Administration Services Console.

➤ To start Administration Services, see "Starting Administration Services" in *Essbase Administration Services Installation Guide*.

Starting and Stopping Applications

When an application is started, Analytic Services loads the application and all associated databases into memory on the Analytic Server. All client requests for data, such as data loads, calculations, reports, and spreadsheet lock and sends, are then handled through the application server, the ESSSVR process. The application server is always started by the Agent.

Multiple applications (ESSSVR processes) can run on Analytic Server concurrently. On Windows, a separate window opens for each ESSSVR process that is running. If an application contains multiple running databases, all databases are managed by the one application server.

When you stop an application, Analytic Services unloads all information and databases from memory on Analytic Server and closes the application server process.

This section contains the following sections:

Starting an Application

When you start an application, the following actions can happen:

- Users can connect to the application.
- The application can respond to commands from the Agent.
- Users can change the settings of the application.
- Data and user security are enabled.
- Each database in the application can start.

➤ To start an application, use any of the following methods:

Tool	Topic	Location
Agent	START *appname*	Enter the command at the command prompt for the Agent in the Analytic Server console window
Administration Services	Starting Applications	*Essbase Administration Services Online Help*
MaxL	**alter system load application**	*Technical Reference*
ESSCMD	LOADAPP or SELECT	*Technical Reference*

This action starts the application and, if you are running on Windows, opens the application server window on the Analytic Server computer.

➤ You can also start an application by completing any of these actions:

- Starting a database within an application, as discussed under "Starting a Database" on page 935.
- Saving an outline to Analytic Server. Opening an outline does *not* start an application.

➤ To control how applications are started, use either of the following methods:

Tool	Topic	Location
Administration Services	Configuring Applications to Start Automatically	*Essbase Administration Services Online Help*
MaxL	**alter application enable autostartup**	*Technical Reference*

When application **startup** (Allow user to start application) is enabled, if an application is stopped and a user attempts to retrieve data from any databases within that application, the application starts on the Analytic Server computer automatically.

When application **autostartup** (Start application when Analytic Services starts) is enabled, users may experience better initial performance when they make requests of databases in that application, because the application and databases are already loaded into memory on the Analytic Server computer.

Stopping an Application

When you stop an application, transactions may be currently running. If you stop an application using any of the following methods, the application does not stop if a calculation or data load is in progress. Instead, Analytic Services displays a message in the Agent console. It is important to stop an application properly. If the application server is not brought down properly, databases within the application may become corrupt.

➤ To stop the application, use any of the following methods:

Tool	Topic	Location
Agent	stop *appname*	Enter the command at the command prompt for the Agent in the Analytic Server console window
Administration Services	Stopping Applications	*Essbase Administration Services Online Help*
MaxL	**alter system unload application**	*Technical Reference*
ESSCMD	UNLOADAPP	*Technical Reference*

If you stop the Agent by closing the server console window or by pressing Ctrl + C, the application stops, and the next time you start the application, Analytic Services rolls back any transactions that were in progress.

See "Rollback with Committed Access" on page 1060 or "Rollback with Uncommitted Access" on page 1062 for a description of the rollback process (undoing all transactions after the last committed transaction).

Stopping an Application Improperly

There are times when stopping the application server process improperly is necessary; for example, if the application server is corrupted and not processing client requests. In this case, stopping the application server by shutting down its corresponding window is the only method of the stopping the application.

➤ To stop the application improperly, use any of the following methods:

Operating system	Instructions to stop the application improperly
Windows	Use any of the following methods: • Perform a Windows operating system End Task. The Windows NT Task Manager does not display process IDs for individual Analytic Services applications—all of the running Analytic Services applications are displayed as undifferentiated ESSSVR processes. This situation prevents you from stopping a single application, in the event that the application freezes. • Click the Close button in the upper-right corner of the application server window. • Kill the process ID by using the process ID number in conjunction with a kill operating system utility. Such a utility is provided with Windows NT Resource Kit and with various other toolkits. You can find the process ID for individual servers in the `essbase.log` file in the *ARBORPATH* directory. When the server starts, a line like the following is displayed in the Analytic Server log: `Application [Sample] started with process id [225]`
UNIX	Kill the ESSSVR process. On UNIX platforms, you can use the ps output to identify individual applications. If an application freezes, you can stop the application by using this command: kill -9 <pid>
Both	On any platform, shut down the Analytic Server computer prior to shutting down the Agent

Starting and Stopping Databases

Starting a database loads the database into memory on the Analytic Server computer. Stopping a database unloads all database information from memory on the Analytic Server computer.

This section contains the following sections:

Starting a Database

When Analytic Services starts a database and loads it to memory, the entire index cache for that database is allocated in memory automatically. The data cache and data file cache are allocated as blocks are requested from Analytic Services clients.

When you start an application, Analytic Services loads the application and its databases into memory on the Analytic Server computer. When you start a database from an application that is not loaded, the application is loaded along with all its related databases.

➤ To start a database, use any of the following methods:

Tool	Topic	Location
Agent	START *appname*	Enter the command at the command prompt for the Agent in the Analytic Server console window.
Administration Services	Starting Databases	*Essbase Administration Services Online Help*
MaxL	**alter application load database**	*Technical Reference*
ESSCMD	LOADDB or SELECT	*Technical Reference*

➤ To configure a database to start automatically when its parent application starts, use either of the following methods:

Tool	Topic	Location
Administration Services	Configuring Databases to Start Automatically	*Essbase Administration Services Online Help*
MaxL	**alter database enable autostartup**	*Technical Reference*

Stopping a Database

Stopping a database unloads all data from memory and commits any updated data to disk.

If a database is stopped and a user attempts to retrieve data from it, the database starts on Analytic Server automatically, without any explicit commands issued.

When you stop a database, transactions may be currently running. If you issue the STOP command in the server console window, an UNLOADDB command in ESSCMD, or **alter application unload database** in MaxL, the database does not stop if a calculation or data load is in progress. Instead, Analytic Services displays a message in the server console window. If you stop the Agent by closing the server console window or by pressing Ctrl + C, the database stops, and the next time you start the database, Analytic Services rolls back any transactions that were in progress.

See "Rollback with Committed Access" on page 1060 or "Rollback with Uncommitted Access" on page 1062 for a description of the rollback process (undoing all transactions after the last committed transaction).

➤ To stop a database, use any of the following methods:

Tool	Topic	Location
Agent	STOP *appname*	Enter the command at the command prompt for the Agent in the Analytic Server console window.
Administration Services	Stopping Databases	*Essbase Administration Services Online Help*

Tool	Topic	Location
MaxL	**alter application unload database**	*Technical Reference*
ESSCMD	UNLOADDB	*Technical Reference*

Managing Security-File Defragmentation

Changing or deleting the following Essbase security entities can cause fragmentation in the security file (essbase.sec): filters, users, groups, applications, databases, substitution variables, disk volumes, passwords, and other Essbase objects. Too much fragmentation in files can slow down security-related performance.

Analytic Services compacts the security file automatically each time the Agent is stopped. You can check the defragmentation status of the security file and, if desired, you can compact the security file without stopping the Agent.

Displaying the Defragmentation Status of the Security File

The defragmentation status of the security file is displayed as a percent.

➤ To display the defragmentation status of the security file, use the following method:

Tool	Topic	Location
MaxL	**display system security file fragmentation_percent**	*Technical Reference*

Compacting the Security File While the Agent is Running

Besides manually compacting the security file, you can use the SECURITYFILECOMPACTIONPERCENT configuration setting to define a percentage of defragmentation that triggers defragmentation automatically.

➤ To compact the security file without stopping the Agent, use either of the following methods:

Tool	Method	Documentation Location
Agent	COMPACT Enter the command at the command prompt for the Agent in the Analytic Server console window	
MaxL	**alter system compact security file**	*Technical Reference*
essbase.cfg	`SECURITYFILECOMPACTIONPERCENT`	*Technical Reference*

Note: Compacting the security file while the Agent is running slows down Agent activity until the operation is completed, which could take a few minutes.

Managing Ports

The Agent enables you to manage ports on Analytic Server. This section contains the following sections:

Viewing a List of Users and Available Ports

The Agent enables you to view a list of all users that are connected to Analytic Server at any given time. Additionally, you can view the total number of ports available, as well as the number of existing connections.

For information about how connections/ports are established in Administration Services, see "About Analytic Services Connections and Ports" in *Essbase Administration Services Online Help*.

➤ To view a list of all users connected to Analytic Server, type the command USERS at the command prompt of the Analytic Server console.

The server console displays the following information:

- The names of all users connected to Analytic Server
- The total number of ports available
- The total number of existing connections
- The application to which each user is connected
- The database to which each user is connected

➤ To view the number of ports installed on Analytic Server, as well as the number of ports in use, use any of the following methods:

Tool	Topic	Location
Agent	PORTS	Enter the command at the command prompt for the Agent in the Analytic Server console window.
Administration Services	Checking Available Ports	*Essbase Administration Services Online Help*
MaxL	**display user**	*Technical Reference*
ESSCMD	LISTUSERS	*Technical Reference*

Specifying Non-Default Port Values

If you wish to change the default port values used by the Agent, you must set one or more of these configuration settings:

- AGENTPORT specifies the port that the Agent uses.
- SERVERPORTBEGIN specifies the first port number the Agent on a single computer tries to use for its first server process.
- SERVERPORTEND specifies the highest value the Agent tries to use for a port when it tries to start a server process. If the value is unavailable, the server process fails.
- PORTINC specifies the value of the increment in between port numbers used by the Agent.

You may wish to change the default for many reasons. Here are two examples:

- The default value is inappropriate for your site because it specifies a port number already in use. To change default values for one installation on a single computer, see "Changing Port Default Values" on page 940.
- You may wish to install a second instance of Analytic Server on a single computer to facilitate testing. Use the configuration settings to assign the second Analytic Server instance to a different port than the first. For detailed instructions, see "Installing Additional Instances of Analytic Server: Windows" on page 942 or "Installing Additional Instances of Analytic Server: UNIX" on page 945.

Changing Port Default Values

If you simply need to change one or more of the default values associated with Agent and server ports, review the *Technical Reference* for instructions about the correct configuration setting to change:

- AGENTPORT specifies the port that the Agent uses.
- SERVERPORTBEGIN specifies the first port number the Agent on a single computer tries to use for its first server process.

- SERVERPORTEND specifies the highest value the Agent tries to use for a port when it tries to start a server process. If the value is unavailable, the server process fails.
- PORTINC specifies the value of the increment in between port numbers used by the Agent.

Viewing Port Statistics

You can enable Analytic Services to log, at a specified interval, the number of ports being used. By analyzing the information in the log, you can monitor port utilization and identify a need for more ports before end users are unable to connect.

To enable Analytic Server to check port use statistics and write those statistics to the Analytic Server log:

Edit the server configuration file `essbase.cfg` to include the setting, PORTUSAGELOGINTERVAL:

```
PORTUSAGELOGINTERVAL n
```

where the value of *n* represents the number of minutes between each check of the number of ports in use. The value of *n* can be any whole number from 1 to 60, with five being the recommended minimum and default value. Analytic Services ignores any portion of a non-whole number. For example, Analytic Services evaluates 2.5 as 2 minutes. Statistics are written to the log immediately after each check.

After adding PORTUSAGELOGINTERVAL to `essbase.cfg`, you need to restart Analytic Server, then view the Analytic Server log file. The log file will look similar to the following output:

```
[Mon Apr 22 00:48:50 2003]Local/ESSBASE0///Info(1056214)
[3] ports in use, [10] ports allowed
```

For more information about the PORTUSAGELOGINTERVAL setting, see the *Technical Reference*.

Managing Administration Server Communication Ports

Administration Server has several configurable communication ports, which are different from Analytic Server ports. For more information, see "About Administration Server" on page 121.

Controlling Query Size and Duration

Users may unintentionally request information that is so large or so complex to retrieve that the query will slow performance or fail to complete properly. To prevent this from occurring, use the following configuration settings to control query size or duration:

- QRYGOVEXECTIME limits the length of time Analytic Server allows a query to run before terminating the query.
- QRYGOVEXECBLK limits the number of blocks a query can access before terminating the query.

These two configuration settings are also referred to as *query governors*.

You can apply these settings to all the applications and databases on Analytic Server, to all the databases on a single application, or to a single database.

For details about how to use these configuration settings, see the *Technical Reference*, in the "essbase.cfg Settings" topic.

Installing Additional Instances of Analytic Server: Windows

You can install multiple instances of Analytic Server on a single Windows computer. For example, you could install a test version and a production version.

➤ To install and configure an additional instance of Analytic Server on a single Windows computer, perform the following tasks:

1. On a computer where you have already installed Analytic Server according to the instructions in the *Essbase Analytic Services Installation Guide*, install the Analytic Server component in a different directory from the first Analytic Server installation. You need not install any of the clients, such as Administration Services.
2. In the *ARBORPATH*`\bin` directory on the Analytic Server computer of the new installation, create or modify the `essbase.cfg` file on the Analytic Server so that it contains these settings:
   - AGENTPORT: The port that this second Analytic Server Agent uses to connect.
   - SERVERPORTBEGIN: The first port that the first Analytic Server process tries to use to connect.
   - SERVERPORTEND: The highest value for a port number that the Analytic Server process can use to connect.
   - PORTINC: The increment between ports. For example, if PORTINC is assigned a value of 5, Analytic Services looks for ports 32700, 32705, and so on up to the value of SERVERPORTEND.

   Use the instructions in the *Technical Reference* to select values for these settings.
3. Create a batch script (`.bat`) that performs these tasks:
   - Sets *ARBORPATH* to the correct value for this installation
   - Adds this path to the *PATH* variable:

     ```
     %arborpath%\bin;%path%
     ```
   - Starts Analytic Server

Note: This installation is a separate installation of Analytic Server. It shares none of the security or objects of any other Analytic Server installation on the same computer.

When multiple instances of Analytic Server are installed on a single machine, you can connect to an Analytic Server agent by specifying the machine name and the agent port number, in the form: *machineName*:*agentPort*.

You can also edit or create the client configuration file to identify the Analytic Server agent port to connect to. The configuration file identifies the default Analytic Server agent port. When you connect to an Analytic Server and specify the machine name only (but no agent port number), you are connected to the Analytic Server agent port specified in the configuration file. You can override the configuration file setting by using the *machineName*:*agentPort* syntax when connecting.

➤ To connect to an Analytic Server when multiple instances of Analytic Server are installed on the same machine, perform either of the following actions:

- Explicitly identify the Analytic Server agent port to connect to.

  When you connect to an Analytic Server agent and are prompted for the hostname, specify the machine name and the Analytic Server agent port in the form: *machineName*:*agentPort*. For example:

  ```
  Enter Host> localHost:9008
  ```

  agentPort is the Analytic Server agent port number, which is specified in the server configuration file for Analytic Server. See step 2 on page 943 for information on editing the server configuration file.

- Create or modify the `essbase.cfg` file on the client machine to identify the Analytic Server agent port to connect to.

  Note: When working with a partitioned database, if the partition's source or target database resides on an Analytic Server where the Agent port is different from the default port, the user's connection to both Analytic Servers must be defined using *servername*::*portnumber*. This rule applies to both Analytic Servers, even if only one of them uses a non-default Agent port.

1. To create or modify the client configuration file, `essbase.cfg`, specify the same settings as you did in step 2 on page 943 for the Analytic Server configuration file. The client configuration file is located on the client computer at `\ARBORPATH\bin`.
2. When you connect to an Analytic Server and are prompted for the hostname, specify the hostname and not the port because the port is read from the configuration file.

   For example:

   ```
   Enter Host> localHost
   ```

   Note: You can override the setting in the client configuration file and connect to a different Analytic Server by using the *machineName*:*agentPort* syntax when prompted for the hostname.

Installing Additional Instances of Analytic Server: UNIX

➤ To install and configure a second instance of Analytic Server on a single UNIX computer, perform these tasks:

1. On the computer where you plan to use two instances of Analytic Server, create a new operating system login ID to run the new Analytic Server. For instructions, refer to your operating system documentation.
2. On a computer where you have already installed Analytic Server according to the instructions in the *Essbase Analytic Services Installation Guide*, install the Analytic Server component in a different directory from the first Analytic Server installation. You need not install any of the clients, such as Administration Services.
3. Assign the correct permissions so the new login ID can be used to run Analytic Server. For instructions, see the *Essbase Analytic Services Installation Guide*.
4. Create the necessary path and library statements in the profile for the new login ID. You may want to copy an existing login ID profile for a login ID that runs the existing Analytic Server installation, and change the values for the new installation.
5. Perform step 2 from “Installing Additional Instances of Analytic Server: Windows.”

6. Repeat previous step for the `essbase.cfg` file on the client.
7. Start the new Analytic Server using the new login ID.

Increasing Agent Connections to Analytic Server

You can use the configuration settings AGENTTHREADS and AGTSVRCONNECTIONS to control the maximum number of threads that the Agent creates to perform the initial connection to Analytic Server.

Increasing the maximum number of possible threads between Analytic Server and the Agent allows more than one user to log on and connect to an application and database at a time.

Note: All requests for information after initial connection and before disconnection are handled by a different set of server threads, whose maximum number is controlled by the SERVERTHREADS configuration parameter.

➤ To increase the maximum possible number of agent threads, perform these tasks:

1. Create or edit the `essbase.cfg` file on Analytic Server to include these settings:

   - AGENTTHREADS *maximum_number_of_threads*

     Maximum number of threads that can be spawned by the Agent. These threads are agent threads, and are used by both the server threads controlled by AGTSVRCONNECTIONS for initial connection and disconnection and for other Agent tasks such as answering a request to list users, list applications, or create new users, for example.

   - AGTSVRCONNECTIONS *maximum_number_of_threads*

     Maximum number of threads that the server can spawn to communicate with Agent for initial connection and disconnection. These threads are unrelated to the server threads whose maximum number is controlled by SERVERTHREADS.

   Keep the value of *maximum_number_of_threads* for AGTSVRCONNECTIONS equal to or less than that value for AGENTTHREADS to avoid wasting resources. Each connection requires

one thread each from server and Agent, so there is no need for higher values for AGTSVRCONNECTIONS. The default value for each setting is 5, the minimum is 1, and the maximum is 500.

2. Save the `essbase.cfg` file.
3. Stop and restart Analytic Server to initialize the changes.

Limiting the Number of User Sessions

You can limit the maximum number of user sessions allowed to connect to Analytic Server at any one time, using the configuration parameter MAXLOGIN. This number includes multiple instances of the same user.

For example, the same user with five open Excel worksheets connected to the same Analytic Server use one port, but five sessions.

You may wish to adjust the value of MAXLOGIN to match computer resources, or to more closely manage concurrent ports and user sessions. A concurrent port is used for each unique combination of client machine, Analytic Server, and login name.

➤ To limit the number of simultaneous user sessions using the configuration setting MAXLOGIN, create or edit the `essbase.cfg` file on Analytic Server to contain this setting:

```
MAXLOGIN maximum_number_of_user_sessions
```

The *maximum_number_of_user_sessions* minimum is 1, maximum is 1048575, and the default value is 1000.

User sessions use the threads whose maximum is controlled by the configuration setting SERVERTHREADS, and are not related to the threads whose maximum is controlled by AGENTTHREADS and AGTSVRCONNECTIONS.

Chapter 42

Managing Applications and Databases

This chapter describes the files that are associated with Analytic Services and describes operations that you can perform to manage Analytic Services applications, databases, and database objects.

This chapter contains the following sections:

Understanding Applications and Databases

An application is a management structure that contains one or more Analytic Services databases and related files. Analytic Services applications and databases usually reside on the Analytic Server. The server computer can store multiple applications.

An Analytic Services database is a data repository that contains a multidimensional data storage array. A multidimensional database supports multiple views of data so that users can analyze the data and make meaningful business decisions.

Files that are related to Analytic Services databases are called *objects*. Database objects perform actions against one or more Analytic Services databases, such as defining calculations or reporting against data. By default, objects are stored in their associated database folder on the server. Some objects can also be saved to a

client computer or to other available network directories. For a detailed description of how Analytic Services stores files, see "Understanding How Analytic Services Files Are Stored" on page 950.

In Analytic Services, the common types of database objects include the following:

- A database outline (a storage structure definition)
- Data sources
- Rules for loading data and building dimensions dynamically (rules files)
- Scripts that define how to calculate data (calculation scripts)
- Scripts that generate reports on data (report scripts)
- Security definitions
- Security filters
- Linked reporting objects
- Partition definitions

Some of these objects are optional, such as calculation scripts, filters, and linked reporting objects.

For a complete description of each database object, see "Understanding Database Objects" on page 127.

Understanding How Analytic Services Files Are Stored

In order to manage applications and databases, you need to know how Analytic Services stores server, application, database, and database object files. In particular, there are a few key directories that you should know about.

These directories are created under the root directory of the Analytic Services installation (the directory path named by the value of *ARBORPATH*):

- An `App` directory stores Analytic Services application files as they are created. Each application is stored in a subdirectory under the `App` directory (for example, `\essbase\App\Sample`).

  Each database in an application is stored in a subdirectory under the application subdirectory (for example, `\essbase\App\Sample\Basic`). Database objects, such as outlines, calculation scripts, report scripts, rules files, and data sources, are typically stored on the server in the *appname* or *dbname* directory, on a client computer, or on a network.

  See Table 39 for a list of application and database files.

- The `bin` directory contains the Analytic Services software. See Table 38 for a list of files stored in this directory.
- The `client` directory contains client-based applications and databases. This folder is created during installation of Spreadsheet Add-in or the Runtime Client.
- The `docs` directory is created if you choose to install online HTML or PDF documentation. This directory contains numerous subdirectories and files.
- The `eas` directory is created if you installed Administration Services on the same computer as Analytic Services. For more information about the contents of this directory, see *Essbase Administration Services Installation Guide*.
- The `locale` directory contains the character-set files necessary for all languages that are supported by Analytic Services, including English.
- If you installed SQL Interface, ODBC-driver documentation is located in the `odbcdocs` directory in PDF format.

Note: On Windows platforms, these directory names may appear with different case.

For more information about all directories created on the server and for information about platform differences, see the *Essbase Analytic Services Installation Guide*.

Server Software File Types

This table lists the types of Analytic Services files that are stored in the `\essbase\bin` directory:

Table 38: Analytic Services File Types in the \essbase\bin Directory

File Extension	Description
bak	Backup of security file
bnd	Microsoft ODBC file for SQL Interface installation using a DB2 database
cfg	Analytic Server configuration file
cnt	Online help contents file
cpl	Microsoft ODBC driver for Windows platforms
dll	Microsoft Windows Dynamic Link Library
eqd	Query Designer files
exe	Executable file
hlp	Online help file
lck	Lock file
lic	License information file for ODBC
pl	Sample Perl script
pm	Perl Module
mdb	Message database file
sec	Security file
sl	HP-UX shared library file
so	Solaris shared library file
xll	Spreadsheet Add-in for Microsoft Excel

Application and Database File Types

The following table lists the file types that Analytic Services uses to store applications, databases, and their related objects.

Table 39: Analytic Services File Types for Applications and Databases

File Extension	Description
alg	Spreadsheet audit historical information
apb	Backup of application file
app	Application file, defining the name and location of the application and other application settings
arc	Archive file
atx	Spreadsheet audit transaction
chg	Outline synchronization change file
csc	Analytic Services calculation script
db	Database file, defining the name, location, and other database settings
dbb	Backup of database file
dbf	dBASE data file
ddb	Partitioning definition file
ddm	Temporary partitioning file
ddn	Temporary partitioning file
esm	Analytic Services kernel file that manages pointers to data blocks, and contains control information that is used for database recovery
esr	Temporary database root file
esn	Temporary Analytic Services kernel file
ind	Analytic Services index file
inn	Temporary Analytic Services index file
log	Server or application log
lro	Linked reporting object file that is linked to a data cell

Table 39: Analytic Services File Types for Applications and Databases (Continued)

File Extension	Description
lst	Cascade table of contents or list of files to back up
mdx	dBASE multiple index file
mxl	MaxL script file (saved in Administration Services)
ocl	Database change log
ocn	Incremental restructuring file
oco	Incremental restructuring file
olb	Backup of outline change log
olg	Outline change log
otl	Analytic Services outline file
otm	Temporary Analytic Services outline file
otn	Temporary Analytic Services outline file
oto	Temporary Analytic Services outline file
pag	Analytic Services database data (page) file
pan	Temporary Analytic Services database data (page) file
rep	Analytic Services report script
rul	Analytic Services rules file
scr	Analytic Services ESSCMD script
sel	Saved member select file
tct	Analytic Services database transaction control file that manages all commits of data and follows and maintains all transactions
tcu	Temporary database transaction control file
trg	Trigger definition file.XML (Extensible Markup Language) format.
txt	Text file, such as a data file to load or a text document to link as a linked reporting object

Table 39: Analytic Services File Types for Applications and Databases (Continued)

File Extension	Description
xcp	Exception error log
xls	Microsoft Excel spreadsheet file

API File Types

The following table lists the types of Analytic Services files that are stored in the *\ARBORPATH*\api sub-directories:

Table 40: Analytic Services File Types in the api Directory

File Extension	Description
a	UNIX static library file
bas	Microsoft Visual Basic program source file, containing header definitions for the Analytic Services API
h	C or C++ header file, containing header definitions for the Analytic Services API
lib	C or C++ program library
np	Named Pipes network library
tcp	TCP/IP network library

Managing Applications, Databases, and Database Objects

This section explains how to manage applications, databases, and database objects:

- "Using the File System to Manage Applications and Databases During Backup" on page 956
- "Monitoring Applications" on page 957

- "Using Analytic Services to Manage Applications and Databases" on page 957
- "Using Analytic Services to Manage Objects" on page 962

For a description of Analytic Services applications, databases, and database objects, see "Understanding Applications and Databases" on page 126 and "Understanding Database Objects" on page 127.

Using the File System to Manage Applications and Databases During Backup

You should not use the platform file system to copy, move, rename, or delete applications and databases. When an application or database is altered through the file system, the Analytic Services security file is unable to recognize the changes. This situation creates a mismatch between what actually exists on the hard drive and what exists according to Analytic Services.

CAUTION: Do not move, copy, modify, or delete any of these files—`ess`*`n`*`.ind`, `ess`*`n`*`.pag`, *`dbname`*`.ind`, *`dbname`*`.esm`, *`dbname`*`.tct`. Doing so may result in data corruption.

The only time the file system should be used to manage applications and databases is during the backup process, where the entire directory for an application or database is copied and stored elsewhere. For a comprehensive discussion of backups, see Chapter 47, "Backing Up and Restoring Data."

Certain application and database files can be successfully managed through the file system:

- Rules files for dimension builds and data loads (`.rul`)
- Data load or dimension build files
- Calculation scripts (`.csc`)
- Report scripts (`.rep`)
- MaxL scripts (`.mxl` or any extension)

To copy or move an outline file (`.otl`), you must use Administration Services. For instructions, see "Copying Outlines" in *Essbase Administration Services Online Help*.

Monitoring Applications

Each application that is loaded is an open task or process in the operating system. On Windows platforms, the application is displayed in a command-line window. On UNIX platforms, the application server is a child process of `ESSBASE`. When the application starts, ESSBASE starts the `esssvr` process. For more information, see "Starting an Application" on page 931.

On Windows platforms, when an application starts, a new icon is displayed in the task bar. You can double-click the icon to view the server window.

Analytic Server records application-level activities in an application log. For more information, see "Using Analytic Services Logs" on page 977.

➤ To view application activities as they occur, use either of the following methods:

Tool	Instruction
On Windows platforms, use the application-process window	Select the command-line window that bears the name of the application.
UNIX	`tail -f logfile`

Using Analytic Services to Manage Applications and Databases

This section describes managing applications and databases. It contains the following sections:

- "Renaming Databases" on page 961
- "Deleting Databases" on page 961

Viewing Applications and Databases

When you start Administration Services Console, the Enterprise View tree is displayed in the navigation panel. Enterprise View is a graphical tree view of the Analytic Services environment. It displays the Administration Servers and Analytic Servers that you select. Your view of the Analytic Services environment may look different from that of other administrators.

Applications and databases, and their associated objects, are represented as nodes beneath the Analytic Server node. Objects are grouped into container nodes. For example, individual applications are contained in the Applications node, and databases are contained in the Databases container node. If sample applications and databases are installed with Analytic Server, they appear in Enterprise View along with your organization's applications and databases.

For more information about operating on applications and databases from Enterprise View, see "About Enterprise View" in *Essbase Administration Services Online Help*.

➤ To create a new application, see "Creating Applications" in the *Essbase Administration Services Online Help*.

Copying or Migrating Applications

You can copy an application to any Analytic Server to which you have appropriate access. You can copy (migrate) an entire application to another Analytic Server, or you can copy an application on the same Analytic Server. For example, you may need to migrate an entire application from a development server to a production server. Or, you may want to copy an application on the same server for testing or for backup purposes.

Analytic Services copies applications differently depending on whether you are copying to the same Analytic Server or to a different Analytic Server. When you migrate applications, you can select the objects to migrate, such as calculation scripts, report scripts, rules files, custom-defined macros and functions, substitution variables, and filters. You can also specify how user and group security is migrated.

Administration Services provides a Migration Wizard that helps you migrate applications. See "Migration Wizard" in *Essbase Administration Services Online Help*.

➤ To copy an application, use any of the following methods:

Tool	Topic	Location
Administration Services	Copying Applications	*Essbase Administration Services Online Help*
MaxL	**create application as**	*Technical Reference*
ESSCMD	COPYAPP	*Technical Reference*

Renaming Applications

When you rename an application, the application and its associated directory (`essbase\app\`*`appname`*) are renamed. All objects within the application (for example, databases or calculation scripts) with the same name as the application are not renamed. Before you rename an application, see "Rules for Naming Applications and Databases" on page 133.

➤ To rename an application, use any of the following methods:

Tool	Topic	Location
Administration Services	Renaming Applications	*Essbase Administration Services Online Help*
MaxL	**alter application**	*Technical Reference*
ESSCMD	RENAMEAPP	*Technical Reference*

Deleting Applications

When you delete an application, all objects within the application are also deleted. The `\essbase\app\`*`appname`* directory and all files located in the directory are deleted.

➤ To delete an application, use any of the following methods:

Tool	Topic	Location
Administration Services	Deleting Applications	*Essbase Administration Services Online Help*
MaxL	**drop application**	*Technical Reference*
ESSCMD	DELETEAPP	*Technical Reference*

Copying Databases

You can copy a database in an application to any Analytic Server and application to which you have appropriate access. You can copy (migrate) an entire database to another Analytic Server, or you can copy a database on the same Analytic Server. For example, you may need to migrate an entire database from a development server to a production server. Or, you may want to copy a database on the same server for testing or for backup purposes. Analytic Services copies databases differently depending on whether you are copying to the same Analytic Server or to a different Analytic Server. For more information, see "Copying Databases" in *Essbase Administration Services Online Help*.

Administration Services provides a Migration Wizard that helps you migrate applications and databases. See "Migration Wizard" in *Essbase Administration Services Online Help*.

When you copy a database, all files associated with the database, except data files (`.pag` and `.ind`), are copied to the destination application. Before copying, make sure you have enough disk space to contain a full copy of the database and its related files.

➤ To copy a database, use any of the following methods:

Tool	Topic	Location
Administration Services	Copying Databases	*Essbase Administration Services Online Help*
MaxL	**create database as**	*Technical Reference*
ESSCMD	COPYDB	*Technical Reference*

Renaming Databases

When you rename a database, the database and its associated directory (`essbase\app\`*`appname`*`\`*`dbname`*), and the outline file (`.otl`) are renamed. All other objects in the database (for example, calculation scripts) with the same name as the database are not renamed.

➤ To rename a database, use any of the following methods:

Tool	Topic	Location
Administration Services	Renaming Databases	*Essbase Administration Services Online Help*
MaxL	**alter database**	*Technical Reference*
ESSCMD	RENAMEDB	*Technical Reference*

Deleting Databases

When you delete a database, all objects within the database are also deleted. The `\essbase\app\`*`appname`*`\`*`dbname`* directory and all files located in the directory are deleted.

➤ To delete a database, use any of the following methods:

Tool	Topic	Location
Administration Services	Deleting Databases	*Essbase Administration Services Online Help*
MaxL	**drop database**	*Technical Reference*
ESSCMD	DELETEDB	*Technical Reference*

Using Analytic Services to Manage Objects

This section describes copying, renaming, and deleting objects, such as outlines, calculation scripts, report scripts, rules files, and data sources:

For descriptions of Analytic Services database objects, see "Understanding Database Objects" on page 127.

CAUTION: The only time the file system should be used to manage applications is during the backup process, where the entire directory for an application or database is copied and stored elsewhere.

Copying Objects

You can copy any database object, except an outline, to another application, database, server, or client location. For instructions on copying outlines, see "Creating and Editing Outlines" on page 140.

➤ To copy an object, use any of the following methods:

Tool	Topic	Location
Administration Services	Topics on copying the specific object; for example, Copying a Rules File	*Essbase Administration Services Online Help*
MaxL	**alter object**	*Technical Reference*
ESSCMD	COPYOBJECT	*Technical Reference*

Renaming Objects

You can rename any object, except an outline. An outline always has the same name as the database, so you need to rename the database to rename the outline.

➤ To rename an object, use any of the following methods:

Tool	Topic	Location
Administration Services	Topics on renaming the specific object; for example, Renaming a Rules File	*Essbase Administration Services Online Help*
MaxL	**alter object**	*Technical Reference*
ESSCMD	RENAMEOBJECT	*Technical Reference*

Deleting Objects

You can delete any object, except an outline. An outline is a required part of a database, so you need to delete the database to delete the outline.

Tool	Topic	Location
Administration Services	Topics on deleting the specific object; for example, Deleting a Rules File	*Essbase Administration Services Online Help*
MaxL	**drop object**	*Technical Reference*
ESSCMD	DELETE command for the object to delete	*Technical Reference*

Locking and Unlocking Objects

Analytic Services uses a check-out facility for database objects to ensure that only one user modifies an object at one time. This section describes how to lock and unlock objects, with the exception of outlines. For information about locking outlines, see "Locking and Unlocking Outlines" on page 142.

Note: Locking objects is not the same as locking data blocks. The Analytic Services kernel handles locking for data blocks, but not for objects. See "Data Locks" on page 1054 for information about locking data blocks.

By default, whenever you open a database object, Analytic Services prompts you to lock the object. You can change this default behavior for some objects; see "Setting Analytic Services Default Options" in *Essbase Administration Services Online Help*.

When an object is locked, Analytic Services does not allow other users to save over, rename, or delete the object. You can open a locked object and edit it, but you cannot save over the existing object. If you want to save changes made to a locked object, save the modified object to a different location. You can execute and copy objects that are locked.

Unlocking Objects

You can unlock objects according to your permissions. In Administration Services, you can view all object locks for an Analytic Server, application, or database.

➤ To unlock an object, use any of the following methods:

Tool	Topic	Location
Administration Services	Locking and Unlocking Objects	*Essbase Administration Services Online Help*
MaxL	**alter object**	*Technical Reference*
ESSCMD	UNLOCKOBJECT	*Technical Reference*

Migrating Applications Using Administration Services

Using Administration Services, you can migrate applications to any Analytic Server to which you have appropriate access, regardless of platform. For example, you may need to migrate an application from a development server to a production server. When you migrate applications, you can select the objects to migrate, such as calculation scripts, report scripts, rules files, custom-defined macros and functions, substitution variables, and filters. You can also specify how user and group security is migrated.

➤ To migrate an application, see "Copying Applications" in the *Essbase Administration Services Online Help*.

Porting Applications Across Platforms

Analytic Services runs on multiple platforms, including Windows and UNIX. For a list of supported platforms and information on how to install and configure Analytic Services on each platform, see the *Essbase Analytic Services Installation Guide*.

After you create an application, you may want to port the application to a server that runs a different operating system. This section describes how to port an application to another Analytic Services computer.

Porting Analytic Services applications across servers involves these steps:

1. Identifying compatible files
2. Checking file names
3. Transferring compatible files
4. Reloading the database

Identifying Compatible Files

If you are porting an Analytic Services application to a server that uses a different operating system, you need to identify which Analytic Services files are compatible with the new operating system.

The following file types are compatible between operating systems:

- Text files. The Analytic Services text files are calculation scripts (`.csc`) and report scripts (`.rep`), and any MaxL or ESSCMD scripts you have developed. Also, data files can be text files.
- Rules files. These files are binary files, but they are compatible between operating systems. Rules files have the extension `.rul`.
- Outline files. These files are binary files, but they are compatible between operating systems. Outline files have the extension `.otl`.

The following file types are incompatible between operating systems and need to be redefined or reloaded on the new server:

- Database files with the extensions `.db` and `.dbb`
- Data files with the extension `.pag`
- Index files with the extension `.ind`
- Security files with the extension `.sec`
- Application files with the extensions `.app` and`.apb`
- Analytic Services Kernel files with the extension `.esm`

Note: If you are using the Linked Reporting Objects feature, you need to relink any files or cell notes on the new server. For a comprehensive discussion of how linked reporting objects are used, see Chapter 11, "Linking Objects to Analytic Services Data."

Checking File Names

When transferring files to a UNIX system, you need to be aware of the case of file names. UNIX is a case-sensitive operating system, and files are recognized only if they have the correct case. For example, in certain MaxL and ESSCMD operations, you need to specify a file name, and the file name must be entered with the correct case.

The Analytic Services system files use the following naming conventions on UNIX systems:

- Executable files have no extension and are uppercase (for example, `ESSBASE`, `ESSCMD`).
- Static library files have the file extension `.a` and are in lowercase (for example, `libessnet.a`).

- Shared library files have the file extension `.sl` on HP-UX, `.so` on Solaris, and `.a` on AIX. These file names are in lowercase (for example, `libesscur.sl`).
- Security files have the file extension `.sec` and are in lowercase (for example, `essbase.sec`).
- Message database files have the file extension `.mdb` and are in lowercase (for example, `essbase.mdb`).
- Online help files have the file extension `.hlp` and are in lowercase (for example, `esscmd.hlp`).

Analytic Services files on UNIX systems are capitalized with *proper* case—the first letter is uppercase, and the remaining letters are lowercase. The following table gives examples of names for different file types:

Table 41: File Naming Examples for UNIX

File Type	Example
Database files	`Mydb.db`
Data files	`Mydb.pag`
Index files	`Mydb.ind`
Outline files	`Mydb.otl`
Rules files	`Atlanta.rul`
Data files to load	`Atlanta.txt`
Calculation scripts	`Mycalc.csc`
Report scripts	`Myrepo.rep`
Archive files	`Mydb.arc`
Application logs	`Myapp.log`

Note: The application name is an exception to the above rule. The application name can be in lower case.

Table 42 lists several examples of valid and invalid file names on UNIX systems:

Table 42: Valid and Invalid File Names on UNIX

Valid File Names	Invalid File Names
Model.csc	MODEL.CSC
Monthly.rep	Monthly.Rep
Forecast.otl	forecast.otl
Actuals.rul	AcTuAlS.rUl
My_File.txt	My_File.Txt

Note: Analytic Services does not allow long file names for applications, databases, calculation scripts, reports, and other database files. All file names for objects you create must conform to the Windows 8.3 convention.

Transferring Compatible Files

If two servers are connected, you can create the application and database directories on the new server and use either FTP (File Transfer Protocol) or Administration Services to transfer the compatible application files. If the servers are not connected, you need to redefine server information on the new server before reloading the database.

Using FTP to Transfer Files

Using FTP, you can transfer files directly between operating systems. You should transfer only the files that are compatible between operating systems, and you should transfer the files in binary mode.

If you have files with the wrong case on a UNIX server, Administration Services can see these files but cannot open them. After you use FTP to transfer files, you should rename the files on the server to ensure that they are capitalized with proper case. Alternatively, you can use FTP to rename the file when you transfer the file:

```
ftp>put oldfile Newfile
```

Using Administration Services to Transfer Files

Using Administration Services, you can transfer files from the client computer to the server in the following ways:

- As part of an application migration. See "Migration Wizard" in the *Essbase Administration Services Online Help*.
- As part of a database migration. See "Copying Databases" in the *Essbase Administration Services Online Help*.
- One object at a time. See the topic for the individual object, for example, "Copying Rules Files," in the *Essbase Administration Services Online Help*.

Redefining Server Information

If the server you are porting to is not connected to the existing server, you need to redefine some information on the new server.

➤ To redefine server information, follow these steps:

1. To create users and specify their permissions, use Administration Services on the new server. For a description of methods, see "Granting Permissions to Users and Groups" on page 840.
2. To create the applications and databases that you want to port, use Administration Services on the new server. For a comprehensive discussion of how to create applications and databases, see Chapter 7, "Creating Applications and Databases."
3. Copy the outline files (`.otl`) for the databases that you want to port from the old server to the same directory location on the new server. Make sure the application is not running while you copy these files. See "Creating and Editing Outlines" on page 140.
4. Copy compatible files from the old server to the new server. For lists of compatible and incompatible files, see "Identifying Compatible Files" on page 965.
5. Reload the database. For a review of the reload process, see "Reloading the Database" on page 970.

Reloading the Database

Database files, such as `.db`, `.pag`, `.esm`, and `.ind`, are not compatible between operating systems. If you port an application to a server on a different operating system, you need to repopulate the database by reloading the data from a data file and a rules file (if applicable). One way you can reload is to export the data to a text file, transfer the text file to the new server, and then use the text file to load data. After the load is complete, calculate the new database.

Chapter 43

Monitoring Data, Applications, and Databases

This chapter provides information on using triggers to monitor data changes in a database and detailed information on using logs to monitor server, application and database activities.

This chapter includes the following topics:

Monitoring Data Changes Using Triggers

The triggers feature provided by Essbase Analytic Services enables efficient monitoring of data changes in a database.

If data breaks rules specified in a trigger, Analytic Services can log relevant information in a file, or, for some triggers, can send an e-mail alert (to a user or system administrator). For example, you might want to notify the sales manager if, in the Western region, sales for a month fall below sales for the equivalent month in the previous year.

There are two types of triggers: on-update triggers and after-update triggers. On-update triggers are activated during the update process, when the block containing the data referenced by the trigger is brought into memory. After-update triggers are activated after the update transaction is complete.

Note: On-update triggers are supported only on block storage databases.

This topic contains the following subtopics:

Administering Triggers

To administer triggers, a user must have Database Designer security privilege. Analytic Services monitors and potentially activates triggers during the following activities:

- Data load
- Calculation
- Lock and send from Essbase Spreadsheet Add-in (does not apply to aggregate storage databases)

Analytic Services does not activate triggers during a database restructure.

➤ To create, change, display, and delete triggers, use any of the following methods:

Tool	Topic	Location
Administration Services	Creating Triggers Editing Triggers Viewing Triggers Enabling and Disabling Triggers Viewing Trigger Spool Files Deleting Triggers	*Essbase Administration Services Online Help*
MaxL	**create trigger** **create or replace trigger** **alter trigger** **display trigger** **drop trigger**	*Technical Reference*

You can see information on enabled and disabled triggers in the application log file when you start Essbase Analytic Server.

Creating On-Update Triggers

When creating on-update triggers, consider the following information:

- You must use a symmetric WHERE clause when defining the area specification for a trigger. See the MDX documentation in the MaxL section of the *Technical Reference*.
- To enable Analytic Services to send e-mail alerts, you must have Java Virtual Machine (JVM) installed on your system.

  Note: E-mails related to Unicode-mode applications are encoded in UTF-8 and require a UTF-8-capable e-mail reader.

- You cannot define a trigger that requires data from the following Analytic Services features:
  - Dynamic Calc
  - Hybrid analysis
  - Partitioning
- You can specify whether all trigger data values are stored in the spool file or whether only the most current values are stored (for example, use the a log_value parameter on the MaxL **create trigger** statement or **create and replace trigger** statement). If the log_value parameter is set to ON, both the new value and old value are logged to the spool file. If the log_value parameter is set to OFF, values are not logged to the spool file. The log_value parameter is active only for data load and lock-and-send activities.

Consider data security when sending e-mail alerts. When Analytic Services activates a trigger and sends an e-mail, it cannot check whether the e-mail recipient is authorized to see the data referenced by the trigger condition.

Avoid referencing a sparse dimension member in a trigger condition. When Analytic Services executes a trigger on one data block, it has to read another data block that may or may not be updated. Depending on the state of the second block, Analytic Services may activate the trigger in error.

The following example is based on the Sample Basic database. Assume that East and West are sparse dimensions and that the following statement defines the trigger condition:

```
FIX (East, West, Sales, Jan, Actual, Cola)
IF ((Sales->East + Sales->West) > 20)
EMAIL sales@hyperion.com
```

Assume that the following Sales data values are stored for East and West:

- Sales->East = 5
- Sales->West = 6

Now assume that a user does a lock and send request to update the data values:

- Sales->East = 15
- Sales->West = 3

When Sales->East is updated to 15, temporarily the database contains the following values:

- Sales->East = 15
- Sales->West = 6

So Analytic Services activates the trigger because 15+6 > 20. Subsequently Analytic Services updates Sales->West to 3. Now the data does not meet the trigger condition because 15+3 < 20. However, Analytic Services has already activated the trigger.

When a data load is followed by a calculation, if both the loaded data and the calculated data meet the trigger condition, Analytic Services activates the trigger twice, once on the data load and once on the calculation.

Creating After-Update Triggers

When creating after-update triggers, consider the following information:

- You must use a symmetric WHERE clause when defining the area specification for a trigger. See the MDX documentation in the MaxL section of the *Technical Reference*.
- After-update triggers send information to a spool file.The most recent value is logged to the spool file. E-mail alerts are not supported.
- You cannot define a trigger that requires data from the Analytic Services hybrid analysis feature.

Effect of Triggers on Performance and Memory Usage

Depending on the number of enabled triggers in a database, there may be a small decrease in performance of calculation and data load. You can control the maximum amount of memory used by the triggers feature by specifying the TRIGMAXMEMSIZE configuration setting in the `essbase.cfg` file. By default, TRIGMAXMEMSIZE is set to 4096 bytes. Choosing to log information in a file, rather than sending e-mail, may improve calculation and data load performance by reducing network traffic.

Trigger Examples

The following examples are based on the Sample Basic database.

Note: You cannot define a trigger that requires data from hybrid analysis members. You cannot define an on-update trigger that requires data from Dynamic Calc members or from 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)
- Eastern region (Market dimension member East)

When the member being calculated is Jan and when the Actual, Sales value of Colas for January exceeds 20, the example sends an e-mail to two e-mail accounts.

```
create trigger Sample.Basic.Trigger_Jan_20
where "(Jan,Sales,[100],East,Actual)"
when Jan > 20 and is(Year.currentmember,Jan) then
mail ([Docs.Company.com],[trgsales@company.com],
   [inventory@company.com],
[Mail sent by 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, and March (the children of Year dimension member Qtr1)
- Colas (Product dimension member 100)
- Eastern region (Market dimension member East)

When the member being calculated is Jan, Feb, or Mar and when the Actual, Sales value of Colas for the month 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. On subsequent trigger activations, both old and new log values are retained in the log files.

```
create or replace trigger Sample.Basic.Trigger_Qtr1_Sales
log_value on
Where "(crossjoin(Qtr1.children, {(Measures.Sales, [100],
  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)
- Eastern region (market East)
- January, February, and March (the children of Qtr1)

The trigger is activated after the update action is complete. If the inventory of Colas in the eastern region falls below 500,000, the example logs an entry in the file Inventory_East.

```
create after update trigger Sample.Basic.Inventory_east
where "(crossjoin ({children([Qtr1])},
{([Market].[East], [Product].[100],
  [Inventory].[Ending Inventory])}))"
when [Ending Inventory] < 500000 then
  spool Inventory_East
end;
```


Using Analytic Services Logs

This topic describes the logs that Analytic Server creates to record information about server, application, and database activities. Table 43 briefly describes each log:

Table 43: Summary of Logs

Type of Log	Location of Log	Information Included
Analytic Server log	*ARBORPATH*\essbase.log	Server activities and errors
Application log	*ARBORPATH*\app*application_name**application_name*.log	Application activities and errors
Query log	*ARBORPATH*\app*appname**dbname**dbname*00001.qlg	Query patterns of Essbase database retrievals
Outline change log	*ARBORPATH*\app*application_name**database_name**database_name*.olg	Changes to the outline

Table 43: Summary of Logs

Type of Log	Location of Log	Information Included
Exception log	One of these locations: *ARBORPATH*\log00001.xcp *ARBORPATH*\app\log00001.xcp *ARBORPATH*\app*application_name*\log00001.xcp *ARBORPATH*\app*application_name**database_name*\log00001.xcp	Errors that result when Analytic Server stops abnormally
Dimension build and data load error logs	One of these locations (See Table 54 on page 1017): *ARBORPATH*\client\dataload.err *ARBORPATH*\app*application_name**application_name*.log	Errors from a dimension build or a data load

This topic describes the information written to a log and explains how you can use that information to maintain, tune, or troubleshoot Analytic Server.

This topic includes the following subtopics:

- "Analytic Server and Application Logs" on page 979
- "Using Analytic Server and Application Logs" on page 993
- "Using Application Log to Monitor Memory Use" on page 1001
- "Implementing Query Logs" on page 1002
- "Understanding and Using the Outline Change Log" on page 1002
- "Understanding and Using Exception Logs" on page 1007
- "Understanding and Using Dimension Build and Data Load Error Logs" on page 1016

For information about Essbase Administration Services logs, see "About the Administration Server Log" in *Essbase Administration Services Online Help*.

Analytic Server and Application Logs

Analytic Server writes activities that occur in Analytic Server and application logs. The Analytic Server log is a text file in the *ARBORPATH* directory named `essbase.log`. In a standard installation, for example, the Analytic Server log is `hyperion\essbase\essbase.log`.

Each application on Analytic Server has its own application log. An application log is a text file in the *`ARBORPATH`*`\app\`*`application_name`* directory named *`application_name`*`.log`. For the Sample Basic database, for example, the application log is `hyperion\essbase\app\sample\sample.log`. Information in application logs can help you to pinpoint where and why an error occurred.

The following topics describe Analytic Server and application logs:

- "Contents of the Analytic Server Log" on page 979
- "Analytic Server Log Example" on page 981
- "Contents of the Application Log" on page 983
- "Example of an Application Log" on page 985
- "Analytic Server and Application Log Message Categories" on page 991

For information about the actions that you can perform on server and application logs, see "Using Analytic Server and Application Logs" on page 993. For information on viewing or analyzing logs using Administration Services, see "About Log Viewer" or "About Log Analyzer" in *Essbase Administration Services Online Help*. For information about specific error messages, see the *Error Messages Guide* in `\docs\errmsgs` in the Analytic Services installation.

Contents of the Analytic Server Log

Analytic Server writes activities that occur on Analytic Server in *`ARBORPATH`*`\essbase.log`. Use `essbase.log` to find out more about the activities on Analytic Server. These are some of the activities that you can assess:

- Who performed an operation
- When an operation was performed
- Errors that occurred when an operation was performed or attempted

Table 44 lists the types of actions logged and the information included in the log message. For information about specific error messages, see the *Error Messages Guide* in `\docs\errmsgs` in the Essbase installation.

Table 44: Contents of the Analytic Server Log (essbase.log)

Type of Action	Information Included
Actions performed at the Analytic Server level, such as logging on Analytic Server or setting security	• Username • Date and time • If changing permissions—user and group information and IP address of user • If logging on—the time and date the user last logged on • If logging off—the length of time the user was logged on • If creating or changing an application or database—request to create or open the object; load, connect, and start the application or database to be created; and lock of the object to be created or changed • If creating, changing, or viewing an application or database—requests to list applications and databases and requests to retrieve access information • Shut down or start up requests • Requests to retrieve and delete the Analytic Server log
Actions performed at the application level, such as viewing application settings or viewing and changing custom-defined macros, custom-defined functions, and substitution variables	• Username • Date and time • Requests to retrieve operating system resources, license information, and system-wide configuration • Retrieving and setting global values • If altering substitution variables—request to list and set substitution variables • If altering custom-defined macros—request to list and delete custom-defined macros • Requests to retrieve and delete the application log

Table 44: Contents of the Analytic Server Log (essbase.log) (Continued)

Type of Action	Information Included
Actions performed at the database level, such as creating rules files, outlines, reports, or calculation scripts	• username • Date and time • Requests to retrieve client settings, user, and group information • Requests to list applications and databases • Requests to retrieve access information • Requests to lock objects • Returning objects

Analytic Server Log Example

Figure 219 shows examples of entries written to `essbase.log` when Analytic Server starts. First, the Sample application and the Basic database are loaded. The log includes information about the time the application and database are loaded, the process ID assigned to the application by the operating system, and the startup of the security authentication module.

Note: You can use the process ID to stop the application improperly if you are unable to perform a normal shutdown. See "Stopping an Application Improperly" on page 933.

Figure 219: Startup Messages in the Analytic Server Log

```
[Tue Nov 06 07:54:16 2001]Local/ESSBASE0///Info(1054014)
Database Basic loaded

[Tue Nov 06 07:54:16 2001]Local/ESSBASE0///Info(1051061)
Application Sample loaded - connection established

[Tue Nov 06 07:54:16 2001]Local/ESSBASE0///Info(1054027)
Application [Sample] started with process id [1300]

[Tue Nov 06 07:54:18 2001]Local/ESSBASE0///Info(1054014)
Database Basic loaded

[Tue Nov 06 07:54:23 2001]Local/ESSBASE0///Info(1051134)
External Authentication Module: [LDAP] enabled

[Tue Nov 06 07:54:23 2001]Local/ESSBASE0///Info(1051051)
Hyperion Essbase Analytic Server - started
```

Figure 220 shows a single error. The admin user tried to rename an application using a name that already exists on Analytic Server. The log includes information about the username, the time of the error, and the operation that failed and caused the error.

Figure 220: Error Messages in the Analytic Server Log

```
[Tue Nov 06 08:00:04 2001]Local/ESSBASE0///Info(1051001)
Received client request: Rename Application (from user admin)

[Tue Nov 06 08:00:04 2001]Local/ESSBASE0///Error(1051031)
Application Testing already exists

[Tue Nov 06 08:00:04 2001]Local/ESSBASE0///Warning(1051003)
Error 1051031 processing request [Rename Application] -
disconnecting
```

Finally, Figure 221 shows a shutdown. The log includes information about the name of the application shut down and the time of the shutdown.

Figure 221: Shut Down Messages in the Analytic Server Log

```
[Tue Nov 06 08:00:46 2001]Local/ESSBASE0///Info(1054005)
Shutting down application Sample

[Tue Nov 06 08:00:52 2001]Local/ESSBASE0///Info(1051052)
Hyperion Essbase Analytic Server - finished
```

Contents of the Application Log

Analytic Server writes application activities that occur, such as calculations, on the server in *ARBORPATH*\app*application_name**application_name*.log. For the Sample Basic database, for example, the application log is hyperion\essbase\app\sample\sample.log. For information about specific error messages, see the *Error Messages Guide* in \docs\errmsgs in the Essbase installation.

Use *application_name*.log to find out more about the activities in an application. These are some of the activities you can assess:

- Who performed a specific operation
- When an operation was performed
- Errors that occurred when an operation was performed or attempted
- Information about dimensions and members to aid in optimization
- The name of an object used to execute an operation (such as a calc script or load file used to perform a calculation or a data load) if the object resides on an instance of Analytic Services

Table 45 lists the types of actions logged and the information included in the log message.

Table 45: Contents of the Application Log (application_name.log)

Type of Action	Information Included
Actions performed at the database level, such as loading data, clearing data, or calculating data	• username • Date and time • Application and database name and time • If starting application—loading Java modules, and reading and writing database and application information • If loading databases—information about dimension sizes, dynamic calculation members, blocks, cache sizes, index page size, and I/O information • If starting databases—application and database setup information, including reading free space information, writing database parameters, retrieving state information, writing application and database definition information, retrieving database volumes, and writing database mapping • If loading—the load command, parallel data load information, cells updated, elapsed load time, and the name of the rules file and data file **Note:** Analytic Services supplies the load rule name only if the client making the request (for example, Administration Services) is at the same release level as Analytic Services; for example, Release 7.0. • If calculating—the name of the calculation script used to perform the calculation • If reporting—the name of the report script
Actions performed at the outline level, such as restructuring	• username • Date and time • Information about dimension sizes, dynamic calculation members, blocks, cache sizes, index page size, I/O information, and restructure elapsed time
Actions performed at the spreadsheet level, such as lock and send	• username • Date and time • Action performed

Example of an Application Log

The following topics show example entries in the application log, including a standard startup and shutdown, and an example of the messages logged when an error occurs.

Example 1: Startup Messages in the Application Log

Figure 222 shows all the entries written to *application_name*.log when Analytic Server starts. The log includes information such as the time the application starts, when application and database information is read and written, when the application is ready for login requests, and when the database is loaded.

Figure 222: Initials Startup Messages in the Application Log

```
[Tue Nov 06 08:47:14 2001]Local/Sample///Info(1002035)
Starting Essbase Server - Application [Sample]

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1200480)
Loaded and initialized JVM module

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1019008)
Reading Application Definition For [Sample]

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1019009)
Reading Database Definition For [Basic]

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1019021)
Reading Database Mapping For [Sample]

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1019010)
Writing Application Definition For [Sample]

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1019011)
Writing Database Definition For [Basic]

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1019022)
Writing Database Mapping For [Sample]

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1013202)
```

```
Waiting for Login Requests

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1013205)
Received Command [Load Database]

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1019018)
Writing Parameters For Database [Basic]

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1019017)
Reading Parameters For Database [Basic]
```

After Analytic Server starts, Analytic Server writes information about the dimensions and members in the outline, such as the dimension sizes and dynamic calculation information, to the application log:

Figure 223: Database Outline Messages in the Application Log

```
[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1019012)
Reading Outline For Database [Basic]

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1007043)
Declared Dimension Sizes = [20 17 23 25 5 3 5 3 15 8 6 ]

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1007042)
Actual Dimension Sizes = [20 14 20 25 4 3 5 3 15 8 5 ]

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1007125)
The number of Dynamic Calc Non-Store Members = [8 6 0 0 2 ]

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1007126)
The number of Dynamic Calc Store Members = [0 0 0 0 0 ]
```

Next, Analytic Server writes information about the blocks in the database, including the block size, the number of declared and possible blocks, and the number of blocks needed to perform calculations (you can use this information to estimate the retrieval performance for members of sparse dimensions tagged as Dynamic Calc) to the application log:

Figure 224: Block-Related Messages in the Application Log

```
[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1007127)
The logical block size is [1120]

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1010008)
Maximum Declared Blocks is [575] with data block size of [1700]
```

```
[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1010007)
Maximum Actual Possible Blocks is [500] with data block size
of [192]

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1200481)
Formula for member [Opening Inventory] will be executed in
[CELL] mode

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1012710)
Essbase needs to retrieve [1] Essbase Kernel blocks in order
to calculate the top dynamically-calculated block.
```


Next, Analytic Server writes information about the caches set for each database to the application log, as illustrated in Figure 225.

Figure 225: Cache Messages in the Application Log

```
[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1012736)
The Dyn.Calc.Cache for database [Basic] can hold a maximum
of [2340] blocks.

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1012737)
The Dyn.Calc.Cache for database [Basic], when full, will
result in [allocation from non-Dyn.Calc.Cache memory].

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1019018)
Writing Parameters For Database [Basic]

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1019017)
Reading Parameters For Database [Basic]

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1070013)
Index cache size ==> [1048576] bytes, [1024] index pages.

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1070014)
Index page size ==> [8192] bytes.

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1070081)
Using buffered I/O for the index and data files.

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1070083)
Using waited I/O for the index and data files.

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1019019)
Reading Data File Free Space Information For Database
[Basic]...
```

```
[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1006025)
Data cache size ==> [3145728] bytes, [2048] data pages

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1006026)
Data file cache size ==> [0] bytes, [0] data file pages
```

The final messages logged at startup refer to general database information:

Figure 226: General Database Messages in the Application Log

```
[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1013205)
Received Command [Get Database Volumes]

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1013205)
Received Command [Set Database State]

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1019018)
Writing Parameters For Database [Basic]

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1019018)
Writing Parameters For Database [Basic]

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1013205)
Received Command [Get Database State]

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1013205)
Received Command [Get Database Info]

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1013205)
Received Command [SetApplicationState]

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1019010)
Writing Application Definition For [Sample]

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1019011)
Writing Database Definition For [Basic]

[Tue Nov 06 08:47:15 2001]Local/Sample///Info(1019022)
Writing Database Mapping For [Sample]
```

Example 2: Errors in the Application Log

The following example shows a single error. An unknown member was found in the data load file; the presence of an unknown member caused the load to fail. First, you see the request for the data load, then the error message, and, finally, information messages describing the data values changed by the data load and the data load elapsed time.

```
[Tue Nov 06 08:49:52 2001]Local/Sample///Info(1013210)
User [admin] set active on database [Basic]

[Tue Nov 06 08:49:52 2001]
Local/Sample/Basic/admin/Info(1013091)
Received Command [DataLoad] from user [admin]

[Tue Nov 06 08:49:52 2001]
Local/Sample/Basic/admin/Info(1003040)
Parallel dataload enabled: [1] block prepare threads,
[1] block write threads.

[Tue Nov 06 08:49:52 2001]
Local/Sample/Basic/admin/Error(1003000)
Unknown Item [500-10] in Data Load, [0] Records Completed

[Tue Nov 06 08:49:52 2001]
Local/Sample/Basic/admin/Warning(1003035)
No data values modified by load of this data file

[Tue Nov 06 08:49:52 2001]
Local/Sample/Basic/admin/Info(1003024)
Data Load Elapsed Time : [0.11] seconds

[Tue Nov 06 08:49:52 2001]
Local/Sample/Basic/admin/Info(1019018)
Writing Parameters For Database [Basic]
```

Example 3: Shutdown Messages in the Application Log

The following messages are logged when Analytic Server performs a normal shut down. First information about the database is retrieved, then the database is unloaded, free space information is written, and the server shuts down.

```
[Tue Nov 06 08:50:26 2001]Local/Sample///Info(1013214)
Clear Active on User [admin] Instance [1]

[Tue Nov 06 08:50:34 2001]Local/Sample///Info(1013205)
Received Command [Get Database Info]

[Tue Nov 06 08:50:34 2001]Local/Sample///Info(1013205)
Received Command [Get Database State]

[Tue Nov 06 08:50:34 2001]Local/Sample///Info(1013205)
Received Command [Get Database Volumes]

[Tue Nov 06 08:50:34 2001]Local/Sample///Info(1013205)
Received Command [Get Database State]

[Tue Nov 06 08:50:34 2001]Local/Sample///Info(1013205)
Received Command [Get Database Volumes]

[Tue Nov 06 08:50:34 2001]Local/Sample///Info(1013205)
Received Command [Unload Database]

[Tue Nov 06 08:50:34 2001]Local/Sample///Info(1019018)
Writing Parameters For Database [Basic]

[Tue Nov 06 08:50:34 2001]Local/Sample///Info(1019020)
Writing Free Space Information For Database [Basic]

[Tue Nov 06 08:50:34 2001]Local/Sample///Info(1013207)
RECEIVED SHUTDOWN COMMAND - SERVER TERMINATING
```

Analytic Server and Application Log Message Categories

Table 46 provides error message categories for each error number range that is shown in the first column. When you receive an error message, use this table to identify the Analytic Services component to which the error is related. For information about specific error messages, see the *Error Messages Guide* in `\docs\errmsgs\erhelp.htm` in the Essbase installation.

Table 46: Error Message Categories

Error Message Number Range	Component That Generated the Error
1001000-1001999	Report Writer
1002000-1002999	General server
1003000-1003999	Data load
1004000-1004999	General server
1005000-1005999	Backup, export, or validate
1006000-1006999	Data cache
1007000-1007999	Outline restructure
1008000-1008999	System calls, portable layer, ASD, or Agent
1009000-1009999	Restoring ASCII data
1010000-1010999	Internal (block numbering)
1011000-1011999	Internal (utilities)
1012000-1012999	Calculator
1013000-1013999	Requestor
1014000-1014999	Lock manager
1015000-1015999	Alias table
1016000-1016999	Report Writer
1017000-1017999	Currency
1018000-1018999	Not currently used
1019000-1019999	Database objects
1020000-102999	Spreadsheet extractor

Table 46: Error Message Categories (Continued)

Error Message Number Range	Component That Generated the Error
1021000-1021999	SQL Interface
1022000-1022999	Security
1023000-1023999	Partitioning
1024000-1024999	Query Extractor
1030000-1030999	Application Programming Interface (API)
1040000-1040999	General network
1041000-1041999	Network—Named Pipes
1042000-1042999	Network—TCP
1043000-1049999	Not currently used
1050000-1055999	Agent
1056000-1059999	Not currently used
1060000-1060999	Outline API
106100-1069999	Not currently used
1070000-1070999	Index manager
1071000-1079999	Not currently used
1080000-1080099	Transaction manager
1081000-1089999	Not currently used
1090000-1099999	Rules file processing
1010000-1019999	Not currently used
1100000-1100999	Not currently used
1110000-1119999	Hyperion Web Gateway (HWG)
1120000-1129999	Grid API
1130000-1139999	Miscellaneous
1140000-1149999	Linked Reporting Objects (LRO)
1150000-1159999	Outline synchronization

Table 46: Error Message Categories (Continued)

Error Message Number Range	Component That Generated the Error
1160000-1169999	Outline change records
1170000-1179999	Attributes
1180000-1189999	Showcase
1190000-1199999	Enterprise Integration Services
1200000-1200999	Calculator framework

Using Analytic Server and Application Logs

The following topics describe the actions you can perform on server and application logs.

- "Setting the Type of Analytic Server Messages Logged" on page 994
- "Setting the Type of Application Messages Logged" on page 995
- "Viewing the Analytic Server and Application Logs" on page 997
- "Clearing the Analytic Server and Application Logs Immediately" on page 998
- "Clearing the Analytic Server and Application Logs Upon Restart" on page 999
- "Setting Delimiters in the Analytic Server and Application Logs" on page 999
- "Analyzing Logs with Log Analyzer" on page 1000

For a comprehensive discussion of server and application logs, see "Analytic Server and Application Logs" on page 979. For information about specific error messages, see the *Error Messages Guide* in `\docs\errmsgs\erhelp.htm` in the Essbase installation.

You can also view logs using Administration Services. For more information, see "About Log Viewer" in *Essbase Administration Services Online Help*.

Setting the Type of Analytic Server Messages Logged

By default, the Analytic Server log, *ARBORPATH*\essbase.log, lists the following types of messages:

- Information messages, such as notification of a user action or information about an application or database

  In the following example, the admin user logged out.

  ```
  [Sun Oct 21 16:00:55 2001]Local/ESSBASE0///Info(1051037)
  Logging out user admin, active for 144 minutes
  ```

- Warning messages, such as a notification that an operation was not completed

  Warnings often follow errors. In the following example, the rename operation did not complete due to a previous error in the log.

  ```
  [Fri Nov 02 13:38:14 2001]Local/ESSBASE0///Warning(1051003)
  Error 1051031 processing request [Rename Application] -
  disconnecting
  ```

- Error messages, such as trying to perform an action that Analytic Server cannot perform.

  In the following example, the rename operation failed because the application name already existed.

  ```
  [Fri Nov 02 13:38:14 2001]Local/ESSBASE0///Error(1051031)
  Application Testing already exists
  ```

This table lists the settings that you specify in the `essbase.cfg` file to determine what types of messages Analytic Server writes to the Analytic Server log. If you change an `essbase.cfg` setting, restart Analytic Server to apply the change.

Setting	Definition	For More Information
AGENTLOGMESSAGELEVEL	An `essbase.cfg` setting that determines whether Analytic Server writes all messages, warning messages, or error messages to the Analytic Server log	*Technical Reference*
PORTUSAGELOGINTERVAL	An `essbase.cfg` setting that specifies how often to check the number of ports in use.	*Technical Reference*

Setting the Type of Application Messages Logged

By default, the application log, *ARBORPATH*`\app\`*application_name*`\`*application_name*`.log`, lists the following types of messages:

- Information messages that detail routine actions that Analytic Server performs

  In the following example, Analytic Server writes the amount of time elapsed during a data load to the application log:

  ```
  [Fri Nov 02 13:04:15 2001]
  Local/Sample/Basic/admin/Info(1003024)
  Data Load Elapsed Time : [3.014] seconds
  ```

- Warning messages that list conditions that are not deemed serious by Analytic Server

  In the following example, Analytic Server writes a statement that no data values were changed during a data load to the application log:

  ```
  [Fri Nov 02 12:43:44 2001]
  Local/Sample/Basic/admin/Warning(1003035)
  No data values modified by load of this data file
  ```

- Error messages that describe errors that occurred while performing the task

  Error messages can range from serious, such as a file not loading correctly, to very serious, such as a disk space error that causes Analytic Server to crash. In the following example, Analytic Server writes a statement to the application log indicating that a data value was encountered before all dimensions in the outline were specified:

  ```
  [Fri Nov 02 12:53:32 2001]
  Local/Sample/Basic/admin/Error(1003007)
  Data Value [678] Encountered Before All Dimensions Selected,
  [2] Records Completed
  ```

This table lists settings that you specify in the `essbase.cfg` file and a command that you can use in a calculation script to determine what types of messages Analytic Server writes to the application log. If you change an `essbase.cfg` setting, restart Analytic Server to apply the change.

Table 47: Setting Messages in the Application Log

Setting	Definition	For More Information
LOGMESSAGELEVEL	An `essbase.cfg` setting that determines whether Analytic Server writes all messages, warning messages, or error messages to the application log	*Technical Reference*
TIMINGMESSAGES	An `essbase.cfg` setting that determines whether Analytic Server writes the duration of each spreadsheet and report query to the application log	*Technical Reference*

Table 47: Setting Messages in the Application Log (Continued)

Setting	Definition	For More Information
SSLUNKNOWN	An `essbase.cfg` setting that determines whether Analytic Server writes error messages when it encounters an unknown member name during a spreadsheet operation to the application log	*Technical Reference*
SET MSG	A calculation script setting that determines whether Analytic Server writes the following items to the application log during the duration of the calculation script: • A summary of calculation statistics • Detailed calculation statistics • All messages, warning messages, error messages, or no messages	*Technical Reference*

Viewing the Analytic Server and Application Logs

When you view a log, you are viewing a snapshot. To view an updated version of a log, close and reopen the log. You can choose to view a log from a specific date to the present or to view the entire log.

You must have Supervisor permissions to view Analytic Server logs, and you must have at least Application Designer permissions to view application logs.

➤ To view a server or application log, use either of the following methods:

Tool	Topic	Location
Administration Services	Viewing Logs	*Essbase Administration Services Online Help*
Any text editor		(See the documentation for the text editor.)

Clearing the Analytic Server and Application Logs Immediately

The server and application logs use disk space on the server. Occasionally, you may need to clear entries from a log before it grows too large. Clearing the server log removes all entries in the log, but does not remove the server log itself. Clearing the application log removes all the entries in the application log and deletes the application log itself. Be sure to back up each log before you clear it.

You must have Supervisor permissions to clear server and application logs.

➤ To clear a server or application log immediately, use any of the following methods:

Tool	Topic	Location
Administration Services	Deleting Logs	*Essbase Administration Services Online Help*
MaxL	To clear the Analytic Server log: **alter system clear logfile;** To clear the application log: **alter application** *application_name* **clear logfile;**	*Technical Reference*
ESSCMD	To clear the server log: DELETELOG ""; To delete the application log: DELETELOG "*application_name*";	*Technical Reference*

Note: To clear a server or application log each time the server or application restarts, see "Clearing the Analytic Server and Application Logs Upon Restart" on page 999. To conserve space by limiting the items logged in the Analytic Server log or application log, see "Setting the Type of Analytic Server Messages Logged" on page 994 or "Setting the Type of Application Messages Logged" on page 995.

Clearing the Analytic Server and Application Logs Upon Restart

By default, Analytic Server appends messages to the end of the server and application logs. As described in Table 48, you can set Analytic Server to clear the Analytic Server log each time the server is restarted or the application log each time the application is restarted. If you change the `essbase.cfg` file, restart Analytic Server to apply the change.

Table 48: Clearing the Server and Application Logs upon Restart

Setting	Description	For More Information
CLEARLOGFILE	An `essbase.cfg` setting that, when set to TRUE, clears the Analytic Server log each time Analytic Server restarts and the application log each time the application restarts	*Technical Reference*

Note: To clear a server or application log without restarting the server or application, see "Clearing the Analytic Server and Application Logs Immediately" on page 998. To conserve space by limiting the items logged in the Analytic Server log or application log, see "Setting the Type of Analytic Server Messages Logged" on page 994 or "Setting the Type of Application Messages Logged" on page 995.

Setting Delimiters in the Analytic Server and Application Logs

You can change the symbol used to delimit log entries for server and application logs. Changing the delimiter also affects messages logged in the server console window. By default, Analytic Server uses spaces to delimit fields in a log, as in the following example:

```
[Thu May 10 20:14:46 2001]Local/ESSBASE0///Info(1051051)
Hyperion Essbase Analytic Server - started
```

You can also use tildes, carets, colons, ampersands, or asterisks to delimit the entries in the server and application logs. If possible, choose a delimiter that does not occur frequently in the data, application, or database. The following example shows a log entry delimited by tildes (~):

```
Thu~May~10~20:16:13~2001~Local~ESSBASE0~~~Info~(1051051)~ \\
Hyperion Essbase Analytic Server - started
```

Note: If you set delimiters to anything other than spaces, you can no longer sort the log entries by date in Log Viewer.

This table lists settings that you specify in the `essbase.cfg` file to determine the delimiters that Analytic Server uses in the server and application logs. If you change an `essbase.cfg` setting, restart Analytic Server to apply the change.

Setting	Description	For More Information
DELIMITEDMSG	An `essbase.cfg` setting that, when set to TRUE, adds a tilde (~) between each field in the server and application logs To specify a different delimiter, use the DELIMITER setting.	*Technical Reference*
DELIMITER	An `essbase.cfg` setting that specifies the delimiter used between each field in the server and application logs Analytic Server enables you to use the following characters as log delimiters: tilde (~), the default delimiter; caret (^); colon (:); ampersand (&); and asterisk (*). The DELIMTER setting only works when DELIMITEDMSG is set to TRUE.	*Technical Reference*

Analyzing Logs with Log Analyzer

You can use Log Analyzer to filter, search, and analyze Analytic Server logs and application logs. Based on filters you choose or create, you can view robust graphical charts for a log. An auto-refresh option enables you to monitor log information dynamically.

➤ To use Log Analyzer, see "About Log Analyzer" in *Essbase Administration Services Online Help*.

Using Application Log to Monitor Memory Use

Essbase enables memory management by providing controls on the maximum amount of memory an application can use. Essbase also writes messages to an application log as soon as the application's memory use exceeds 90% of the limit that you set. After the limit is met 100%, Analytic Server stops the application.

To enable Analytic Server to manage application memory use:

1. Create a new memory configuration file, `config.mem`, containing the MEMORYLIMIT setting, or edit an existing memory configuration file.
2. Save the memory configuration file in the *ARBORPATH*`/bin` directory.
3. Start the application. You must restart an application after you change its memory configuration file because the application reads the memory configuration file only once during start up.
4. Monitor the log.

When 90% of the value in the memory configuration file is reached, Analytic Server writes the following warning message to the application log:

```
Application is using [n] bytes of memory. It is more than 90
percent of the limit set in the memory config file.
```

If the application uses 100% of the memory as limited by the memory configuration file, Analytic Server writes the following error message to the application log and stops the application:

```
Application is using [n] bytes of memory, and is requesting a new
memory block of [n] bytes. The memory has exceeded the limit.
```

When the 90% warning messages are displayed in the log, the database administrator may wish to take corrective action:

- Increase the memory limit in the relevant memory configuration file and restart the application if more memory is available.
- Examine cache sizes and reduce the size if possible.
- If neither of these actions is possible or resolves the memory problem, work loads or applications may require additional adjustments or tuning.

For more information about the memory configuration file and MEMORYLIMIT setting, see the *Technical Reference*.

Implementing Query Logs

Query logging provides a way for Essbase administrators to track query patterns of an Essbase database. The query log file tracks all queries performed against the database regardless of whether the query originated from Spreadsheet Add-in or Report Writer. Query logging can track members, generation or level numbers of members belonging to specific generations or levels, and Hybrid Analysis members. Query logging also offers the flexibility to exclude logging of certain dimensions and members belonging to generations or levels. Because the query log file output is an XML document, you can import the log file to any XML-enabled tool to view the log. For details about the query log file structure, refer to querylog.dtd in the *ARBORPATH*`/bin` directory.

To enable query logging, create a query configuration file (distinct from the `essbase.cfg` file) and add to the file the configuration settings that control how query logging is performed.

In the *ARBORPATH*`\App\`*appname*`\`*dbname* directory of Essbase, create a query log configuration file. The configuration file must be named `dbname.cfg`, where *dbname* matches the name of the database. For example, the query log configuration file for Sample Basic is *basic.cfg*. The output query log file is located at, by default, *ARBORPATH*`\App\`*appname*`\`*dbname*`00001.qlg`.

For more information about query logging and how to set up the query log configuration file, see the *Technical Reference*.

Understanding and Using the Outline Change Log

You can set Analytic Server to create an outline change log that saves outline modification information to a text file. You can review the outline change log at any time to see all the changes that have been made to an outline since the log was first created. This information helps you to roll back an outline to a previous version. The outline change log is a text file in the *ARBORPATH*`\app\`*application_name*`\`*database_name* directory named *database_name*`.olg`. For the Sample Basic database, for example, the outline change log is `hyperion\essbase\app\sample\basic\basic.olg`.

The following topics describe the outline change log and the actions that you can perform on it.

Understanding the Contents of the Outline Change Log

You can set Analytic Server to write changes to the outline to *ARBORPATH*\app*application_name*\database_name*database_name*.olg.

This table lists the types of actions written to the outline change log and the information included in the log message.

Table 49: Outline Change Log Contents

Type of Change	Information Included
Add a dimension	• Name of dimension • Type of dimension (dense or sparse) • Dimension tag (if any) • Name of left sibling of dimension (if any) • Level number of dimension • Generation number of dimension
Delete a dimension	Name of dimension
Update a dimension	• Name of dimension • Dimension tag • Type of dimension (dense or sparse) • Level name changes (if applicable) • Generation name changes (if applicable)

Table 49: Outline Change Log Contents (Continued)

Type of Change	Information Included
Rename a dimension	• Old name of dimension • New name of dimension
Move a dimension to a new position	• Name of dimension • Old location, including left sibling of dimension • New location, including left sibling of dimension
Add a member to a dimension	• Name of new member or members • Unary calculation symbol for member • Level number of member • Generation number of member • Status of member (Store, Share) • Member alias (if applicable) • Account type of member (if applicable) • User-defined attributes of member (if applicable) • Calculation formula for member (if applicable)
Update a member of a dimension	• Name of member updated • Member properties that were updated
Rename a member of a dimension	• Old name of member • New name of member
Move a member of a dimension to a new position	• Name of member moved • New location • Names of parent and left sibling in new location

The outline change log program reads outline information from left to right. If you are looking at an outline, the *left sibling* is the sibling directly above (to the left of) the newly added dimension or member. This rule does not apply if the immediately preceding dimension or member is a *parent*. If a newly added (or moved) member is the first child of its parent or if the member is the first dimension in the outline, the outline change log identifies the old location of the dimension or member as None.

Reviewing an Example of an Outline Change Log

When a user makes and saves changes to an outline, Analytic Services writes the change information into the outline change log as a group of entries. Each group of entries begins and ends with identifying information so that you can easily identify each revision, from newest to oldest.

Figure 227 shows one record, indicating that since the outline change log was started, the outline was modified once. First, the outline change log lists the beginning of the change, including the application and database changed; the time of the change; and the user making the change. Next, the outline change log lists the change—a member named 100-50 was added to 100 member of the Product dimension. Finally, the outline log lists the end of the change, including the application and database changed; the time of the change; and the user making the change.

Figure 227: Sample Outline Change Log

```
[Begin Outline Change for Sample/Basic, Sat Nov 03 12:49:31 2001,
By admin]
Number of member changes for dimension "Product" : 1

Added new member "100-50" to "100" :
Left sibling - "100-40"
Status - Store Data
Added alias "Cherry Cola" to alias table "Default"
Unary calc symbol - Add
Level Number - 0
Generation Number - 3
[End Outline Change for Sample/Basic, Sat Nov 03 12:49:32 2001,
By admin]
```

Creating Outline Change Logs

By default, Analytic Server does not create an outline change log. Table 50 lists the setting that you specify in the `essbase.cfg` file to create an outline change log. If you change an `essbase.cfg` setting, restart Analytic Server to apply the change.

Table 50: Creating an Outline Change Log Using essbase.cfg

Setting	Description	For More Information
OUTLINECHANGELOG	An `essbase.cfg` setting that, when set to TRUE, creates an outline change log	*Technical Reference*

Note: During a restructure, Analytic Services holds outline change information in memory until all updates have been made to the outline change log. Turning on the outline change log may, therefore, affect restructure performance. "Conditions Affecting Database Restructuring" on page 1149 discusses other conditions that affect restructure performance.

Viewing Outline Change Logs

Because the outline change log is an ASCII text file, you can view it by opening it in any text editor. The outline change log is located in the *ARBORPATH*`\app\`*application_name*`\`*database_name* directory and is named *database_name*`.olg`. For the Sample Basic database, for example, the outline change log is located in `hyperion\essbase\app\sample\basic\basic.olg`.

Setting the Size of the Outline Change Log

The default size for the outline change log is 64,000 bytes. When the log reaches this size, Analytic Services copies the contents of *database_name*`.olg` to a new file with the `.olb` extension. For example, when `basic.olg` reaches 64,000 bytes the contents are copied to `basic.olb`. After this copy, `basic.olg` is cleared. New log entries are written to `basic.olg`.

Each time the outline change log reaches its maximum file size, Analytic Services writes over the existing *database_name*`.olb` with the current *database_name*`.olg`. Retrieve information from *database_name*`.olb` before *database_name*`.olg` fills up.

The default, minimum, and maximum file sizes for the *database_name*.olb file are the same as the corresponding defaults specified for the *database_name*.olg file. For example, if you change the maximum size of the outline change log to 2 MB, you automatically set the .olb file to the same maximum size.

Table 51 lists the setting that you specify in the essbase.cfg file to set the size of the *database_name*.olg file. If you change an essbase.cfg setting, restart Analytic Server to apply the change.

Table 51: Setting the Outline Change Log Size Using essbase.cfg

Setting	Instructions	For More Information
OUTLINECHANGELOGFILESIZE	An essbase.cfg setting that determines the size of the outline change log, *database_name*.olg	*Technical Reference*

Understanding and Using Exception Logs

When an Analytic Server, an application, or a database shuts down abnormally, Analytic Server sometimes creates an exception log as a text file named log0000*n*.xcp. The following topics describe the server, application, and database exception logs and the actions that you can perform on them.

- "Understanding the Contents of Exception Logs" on page 1007
- "Reviewing an Example of an Exception Log" on page 1010
- "Viewing Exception Logs" on page 1014
- "Overwriting or Retaining Existing Logs" on page 1015

Understanding the Contents of Exception Logs

If an Analytic Server, an application, or a database shuts down abnormally and cannot restart, Analytic Server generates an exception log to help troubleshoot the problem. The location of the exception log depends on the component that shut down abnormally and the amount of information that Analytic Server had available at the time of the abnormal shutdown. Table 53 on page 1015 describes the location of the exception log for each type of abnormal shutdown.

This table lists the sections of the exception log and the information included in each section. If Analytic Server could not retrieve all the information before the shutdown finished, some of the later sections may be blank.

Table 52: Contents of the Exception Log (log00001.xcp)

Section of Log	Information Included
General information	• Date and time • Application and database name • Location of exception log • Process type Use this information to determine which component shut down abnormally, and when it shut down.
Machine registers and stack information	• General registers • Floating point registers • Hex registers • Stack trace • Stack dump Hyperion Technical Support can examine this section of the log to assist in determining why an abnormal shutdown may have occurred.
Application-wide configuration	• Server and application name • Elapsed application time; that is, how long the application was running • List of modules Use this information to determine if the application shut down quickly and that all modules are correct. More information about modules is in the system-wide configuration section of the exception log.
Operating system resources	• System date and time • Elapsed operating system time; that is, how long the operating system was running • Resource information, including CPU type, memory information, swap information, and drive information Use this information to see if it is an operating system problem, such as a lack of memory.

Table 52: Contents of the Exception Log (log00001.xcp) (Continued)

Section of Log	Information Included
System-wide configuration	• Elapsed Essbase time; that is, how long Essbase was running • Essbase release information • Network information • Environment variables • Module information, including module name and release Use this information to make sure that the release is the same for Essbase and each module, and the environment variables are set correctly.
`essbase.cfg` values	Values of all settings in the `essbase.cfg` file Use this information to make sure that Analytic Server is configured correctly.
License information	• Serial number and license expiration date • Number of ports purchased and ports in use • Essbase options enabled • Other Hyperion products enabled Use this information to make sure that the correct options of Essbase are installed and that you have purchased enough ports.
Client request activity	• Server name • Application name • Thread information, including the number of threads • Request information • Detailed information about each thread, including the action it is performing, the database, username, start time, and end time Use this information to see how heavy the load on the server was, based on client requests.

Table 52: Contents of the Exception Log (log00001.xcp) (Continued)

Section of Log	Information Included
File information	• Page file size • Index file size Use this information to determine whether the page file or the index is too large.
Database information	Use this information to make sure that the database is set up correctly.
Database statistics	Use this information to view dimension information and to see characteristics of data blocks in the database.

Reviewing an Example of an Exception Log

The following example is of an exception log. The first section of the log lists general information about the application and database. In this example, the Analytic Server shut down:

```
----- Exception Error Log Begin -----

Current Date & Time:   Sat Nov 24 13:25:13 2001
Process Type:          Server
Application Name:      Sample
Database Name:         Basic
Exception Log File:    C:\HYPERION\ESSBASE\log00001.xcp
Current Thread Id:     1116
Exception Code:        0xC0000005=Access Violation
Exception Flags:       0x00000000=Continuable
Exception Address:     0x002D2249
Exception Parameters:  2
Exception Parameter 0: 0x00000000=Read Violation
Exception Parameter 1: 0x0000220A (Virtual Address)
```

The next section of the log lists register and stack trace information. Hyperion Technical Support can examine this section of the log to assist in determining why an abnormal shutdown may have occurred.

```
----- Machine Registers -----

General Registers:
   EAX=0x00000000  EBX=0x01358008  ECX=0x00002200

Control Registers:
   CS =0x0000001B  EIP=0x002D2249  Flg=0x00010202

Segment Registers:
   DS =0x00000023  ES =0x00000023  FS =0x00000038

Floating Point Registers:
   CWD=0xFFFF027F  SWD=0xFFFF0000  TWD=0xFFFFFFFF

Register Area (Hex):
   00 00 00 00 00 00 00 00 00 00

...continued hexadecimal listings...

Debug Registers:
   DR0=0x2F75C73B  DR1=0x75E07D39  DR2=0x1475FF16
   DR3=0x00000000  DR6=0x0000E00B  DR7=0x00000000
----- Stack -----

Stack Trace:
    0: 0x002D2249
    1: 0x002D202D
...continued stack trace listings...

Stack Dump (Hex):
   (Stack dump truncated from 1397 to 1024 bytes.)
   0x0012EA2C:  00000000 01358008 01358008 FFFFFFFF
   0x0012EA3C:  00002200 002D728C 0012EC6C 002D202D
...continued stack dump listings...
```

The following section of the log lists application information:

```
----- Application-Wide Configuration -----

Server Name:           ASPEN
Application Name:      Sample
Elapsed App Time:      00:00:01:28
Module Count:          6
Module  0:             0x00241000 =
C:\HYPERION\ESSBASE\BIN\ESSSEC.DLL
Module  1:             0x002C1000 =
C:\HYPERION\ESSBASE\BIN\ESSNET.DLL
...continued module listings...
```

The following section of the log lists operating system information. You can determine how much memory is available, how much swap space is used, and how much memory is available on each drive:

```
----- Operating System Resources -----

System Date & Time:    Sat Nov 24 13:25:13 2001
Elapsed OS Time:       02:03:03:10
OS Name & Version:     Windows NT 5.00
CPU Count:             1
CPU Type:              Pentium
Total Physical Memory: 327024 KB (334872576)
Free Physical Memory:  155760 KB (159498240)
Used Physical Memory:  171264 KB (175374336)
Swap Flags:
   Enabled:            Y
   Disabled:           N
   File Found:         Y
   Denied:             N
Swap file(s):          C:\pagefile.sys
Total Swap Space:      467192 KB (478404608)
Free Swap Space:       421528 KB (431644672)
Used Swap Space:       45664 KB (46759936)
Total Drives:          5
Current Drive:         3
Drive  1:
   Drive Name:         C
   Volume Label:
   Drive Type:         Fixed
   File System:        NTFS
   Total Drive Space:  11778448 KB
```

```
  Free Drive Space:   8592548 KB
   Used Drive Space:   3185900 KB
...continued drive listings...
```

The following section of the log lists system configuration information, such as paths or `essbase.cfg` settings:

```
----- System-Wide Configuration -----

Elapsed Essbase Time:  00:00:01:33
Essbase Version:       6.2.0
Essbase Description:   Ess62P0B128
Network Type:          Windows Sockets
Environment Variable:  ARBORPATH = C:\HYPERION\ESSBASE
Environment Variable:  ARBORMSGPATH = C:\HYPERION\ESSBASE\bin
Module Count:          13
Module  0:
   Module Name:        C:\HYPERION\ESSBASE\BIN\ESSUTL.DLL
   Module Version:     6.2.0.1
   Module Description: Ess62P0B128.1
   Module Use Count:   5
...continued module listings...

----- ESSBASE.CFG Configuration Values -----

Configuration Value:   JvmModuleLocation =
C:\Hyperion\Essbase\java\jre13\bin\hotspot\jvm.dll
Configuration Value:   AuthenticationModule = LDAP essldap.dll x
Configuration Value:   OUTLINECHANGELOG = TRUE
```

The following section of the log lists license information (such as a serial number), the Essbase options (such as ports purchased), and Hyperion products purchased (such as Enterprise Integration Services):

```
----- License Information -----

Serial Number:         xxx
License Expiry Date:
Port Count:            10
Ports In Use Count:    0
Limited Use Version:   N
Read-Only SS:          N

...continued Essbase options and Hyperion product listings...
```

The following section of the log lists client activity, such as using Administration Services to view databases or using the Spreadsheet Add-in to view databases:

```
----- Client Request Activity -----

Server Name:           ASPEN
Application Name:      Sample
Total Request Threads: 5
Avail Request Threads: 6
Total Requests:        56
Average Requests:      48.000000
Weighted Average:      7.440000
Statistics Per Minute:
   Current Requests:   48
   Minimum Requests:   48.000000
   Maximum Requests:   48.000000
Thread Count:          5
Thread Id 1444:
   Request Name:        List Objects
   Database Name:       Basic
   User Name:           admin
   Start Time:          Sat Nov 24 13:24:37 2001
   End Time:            Sat Nov 24 13:24:37 2001
...continued thread listings...

----- Exception Error Log End -----
```

Viewing Exception Logs

Because the exception change log is an ASCII text file, you can view it by opening it in any text editor. The location of the exception log depends on the component that shut down abnormally and the amount of information that Analytic Server had available at the time of the abnormal shutdown.

This table describes the location of the exception log.

Table 53: Location of the Exception Log

Component That Shut Down	Location of the Exception Log
Analytic Server	The log is in the *ARBORPATH* directory; for example, `d:\essbase\log00001.xcp`
Application	If the application name is unknown, the log is in the *ARBORPATH*`\app` directory; for example, `d:\essbase\app\log00001.xcp` If the application name is known, the log is in the application directory; for example, if the Sample application shut down abnormally, `d:\essbase\app\Sample\log00001.xcp`
Database	If the database name is unknown, the log is in the application directory; for example, if the Basic database shut down abnormally: `d:\essbase\app\sample\log00001.xcp`. If the database name is known, the log is in the database directory; for example, if the Basic database shut down abnormally, `d:\essbase\app\sample\basic\log00001.xcp`

Overwriting or Retaining Existing Logs

By default, Analytic Server creates one or more new exception logs each time it shuts down abnormally. Subsequent exception logs are numbered sequentially; for example, if `log00001.xcp` exists, the next log is named `log00002.xcp`, the next is `log00003.xcp`, and so on.

You can change a setting in the `essbase.cfg` file to overwrite the existing exception log instead of creating a new log. If you change an `essbase.cfg` setting, restart Analytic Server to apply the change.

Hyperion recommends that you use the default setting of FALSE. An abnormal shutdown may create multiple exception logs and the first log created during the shutdown is often the most descriptive.

Use this table for more information about creating new exception logs:

Setting	Description	For More Information
EXCEPTIONLOG OVERWRITE	An `essbase.cfg` setting that, when set to FALSE, creates a new exception log. When set to TRUE, Analytic Server overwrites the existing exception log.	*Technical Reference*

Understanding and Using Dimension Build and Data Load Error Logs

Analytic Server writes errors that occur during a dimension build or data load in error logs. The log that Analytic Server chooses for errors depends on the operation that you perform, such as a dimension build or a data load, and how you perform it, such as using Administration Services, or MaxL. The following topics describe the location of dimension build and data load errors and the actions that you can perform on dimension build and data load error logs:

- "Understanding and Viewing Dimension Build and Data Load Error Logs" on page 1016
- "Reviewing an Example of a Dimension Build and Data Load Error Log" on page 1017
- "Setting the Maximum Number of Errors" on page 1018
- "Loading Dimension Build and Data Load Error Logs" on page 1018

Understanding and Viewing Dimension Build and Data Load Error Logs

The `dataload.err` log contains errors that occurred during a dimension build or a data load. The logs also contain the records that failed to load. After you fix the errors, you can reload the logs. For instructions, see "Loading Dimension Build and Data Load Error Logs" on page 1018.

Analytic Server writes errors that occur during a dimension build or data load to the following error logs:

Table 54: Location of Dimension Build and Data Load Errors

Operation	Location of Error Log
Dimension build	*ARBORPATH*\client\dataload.err
Data load with a rules file	*ARBORPATH*\client\dataload.err
Data load without a rules file	*ARBORPATH*\app*application_name**application_name*.log

Note: If the data load or dimension build fails and there is no error log, see "Checking Error Logs" on page 412 for a description of possible causes.

To set the location and file name of dimension build and data load error logs, see "Performing a Data Load or Dimension Build for Block Storage Databases" or "Performing a Data Load or Dimension Build for Aggregate Storage Databases" in the *Essbase Administration Services Online Help*.

➤ To view the dimension build and data load error logs, open them in any text editor.

Reviewing an Example of a Dimension Build and Data Load Error Log

Figure 228 shows an entry in a data load log. The 500 member did not exist in the outline, so no data was loaded to it.

Figure 228: Error Message in the Data Load Log

```
\\ Member 500 Not Found In Database
500   500-10   500-10-10
```

To solve this problem, you can perform either of the following actions and then restart the load:

- Perform a dimension build to create the missing member. For references to instructions, see "Performing Data Loads or Dimension Builds" on page 406.

- Add the missing member in Outline Editor. For a discussion of rules and for instructions, see "Adding Dimensions and Members to an Outline" on page 143.

Figure 229 shows an entry in a dimension build log. The 600-20 member is not the parent of the 600 member. Make sure that you use the correct rules file with the correct text file. The record looks like it is for a level (bottom-up) build, but the error message indicates that Analytic Server is trying to perform a generation (top-down) build. When you correct the problem, restart the dimension build.

Figure 229: Error Message in the Dimension Build Log

```
\\Record #2 - Incorrect Parent [600-20] For Member [600] (3307)
600-20-10   600-20   600
```

Setting the Maximum Number of Errors

The default size of the `dataload.err` files is 1,000 records. When the log reaches this size, Analytic Server does not write any other errors that it encounters to the log. The dimension build or data load, however, continues. Any subsequent errors are lost. You can set Analytic Server to write 1 to 65,000 records in the `dataload.err` files. If you change the `essbase.cfg` file, restart Analytic Server to apply the change.

Table 55: Setting the Number of Error Records

Setting	Description	For More Information
DATAERRORLIMIT	An `essbase.cfg` setting that determines the number of records logged in the `dataload.err` log	*Technical Reference*

Loading Dimension Build and Data Load Error Logs

If the dimension build or data load fails, you must determine whether the Isolation Level transaction setting is Committed or Uncommitted. If the Isolation Level transaction setting is Committed, you must restart the data load from the beginning. If the Isolation Level is Uncommitted, you can load only the records that failed by loading the error log. Only reloading the failed records is much faster than reloading every record, including those records that succeeded during the first load.

➤ To reload the error log, perform the following steps:

1. If you load from the server, change the file extension from `.err` to `.txt`. For example, change the `dataload.err` file to `dataload.txt`.

   If you load from the client, you can leave the `.err` extension.

2. Fix the problem that caused the dimension build or data load to fail. Fixing the problem might involve changing the outline, the text in the error log, or the rules file.

   Check to see if the following conditions are true:

   - Can you validate the rules file? See "Validating, Saving, and Printing" on page 386.
   - Is the data source available? See "Verifying That the Data Source Is Available" on page 411.
   - Is the server available? See "Verifying That Analytic Server Is Available" on page 411.

3. Load the error log using the appropriate rules file. For more information, see Chapter 16, "Understanding Data Loading and Dimension Building."

Chapter

44

Managing Database Settings

This chapter introduces you to the Analytic Server Kernel, explains how Analytic Services accesses and stores data, and provides an overview of Analytic Server database settings you can use to optimize performance at the Analytic Server level.

This chapter contains the following sections:

Note: For information about fatal errors in the Analytic Server Kernel, see "Understanding Fatal Error Handling" on page 1351.

Understanding the Analytic Server Kernel

The kernel provides the foundation for a variety of functions of Analytic Server. These functions include data loading, calculations, spreadsheet lock&send, partitioning, and restructuring. The kernel reads, caches, and writes data; manages transactions; and enforces transaction semantics to ensure data consistency and data integrity.

The kernel has the following functions:

- Handles disk storage and caching of Analytic Services files
- Handles data retrieval
- Handles data updates
- Controls input-output functions related to Analytic Services
- Consolidates free space for re-use
- Manages concurrent operations
- Recovers databases after a server crash
- Issues locks
- Manages transactions

The rest of this section explains the two available access modes, and describes how to set the access modes:

- "Understanding Buffered I/O and Direct I/O" on page 1023
- "Viewing the I/O Access Mode" on page 1023
- "Setting the I/O Access Mode for Existing Databases" on page 1024
- "Setting the I/O Access Mode for New and Migrated Databases" on page 1024

Understanding Buffered I/O and Direct I/O

The Analytic Services Kernel uses buffered I/O (input/output) by default, but direct I/O is available on most of the operating systems and file systems that Analytic Services supports. For a list of the supported platforms, see the *Essbase Analytic Services Installation Guide*.

Buffered I/O uses the file system buffer cache.

Direct I/O bypasses the file system buffer cache, and is able to perform asynchronous, overlapped I/Os. The following benefits are provided:

- Faster response time. A user waits less time for Analytic Services to return data.
- Scalability and predictability. Analytic Services lets you customize the optimal cache sizes for its databases.

If you set a database to use direct I/O, Analytic Services attempts to use direct I/O the next time the database is started. If direct I/O is not available on your platform at the time the database is started, Analytic Services uses buffered I/O, which is the default. However, Analytic Services will store the I/O access mode selection in the security file, and will attempt to use that I/O access mode each time the database is started.

Note: Cache memory locking can only be used if direct I/O is used. You also must use direct I/O if you want to use an operating system's no-wait (asynchronous) I/O.

Viewing the I/O Access Mode

Buffered I/O is the default for all databases.

➤ To view which I/O access mode a database is currently using or is currently set to, use any of the following methods:

Tool	Topic	Location
Administration Services	Selecting an I/O Access Mode	*Essbase Administration Services Online Help*
MaxL	**display database**	*Technical Reference*
ESSCMD	GETDBINFO	*Technical Reference*

Setting the I/O Access Mode for Existing Databases

➤ To use direct I/O instead of the default buffered I/O for any database, use any of the following methods:

Tool	Topic	Location
Administration Services	Selecting an I/O Access Mode	*Essbase Administration Services Online Help*
MaxL	**alter database**	*Technical Reference*
ESSCMD	SETDBSTATEITEM	*Technical Reference*

You may also need to increase the size of some caches. See "Sizing Caches" on page 1128 for instructions and recommendations.

Setting the I/O Access Mode for New and Migrated Databases

To use direct I/O instead of the default buffered I/O for all databases migrated from an earlier release and for newly created databases, follow these steps:

1. Add the configuration setting DIRECTIO to the Analytic Server `essbase.cfg` file, and set the value to TRUE. See the *Technical Reference* for instructions.
2. Consult the *Essbase Analytic Services Installation Guide*, in PDF format, in the `/ARBORPATH/docs/pdf` directory for information and instructions to help you make the transition. *ARBORPATH* is the Analytic Services install directory; by default this is `/hyperion/essbase`.
3. Consult the cache sizing information in "Sizing Caches" on page 1128.

Understanding Kernel Components

The kernel contains components that control all aspects of retrieving and storing data:

- The Index Manager finds and tracks the location of requested data. See "Index Manager" on page 1025 for details.
- The Allocation Manager, part of the Index Manager, allocates space and manages some file operations. See "Allocation Manager" on page 1026 for details.
- The Data Block Manager retrieves the data pointed to by the index and stores the data. See "Data Block Manager" on page 1027 for details.
- The LRO Manager handles retrieval and storage of linked reporting objects (LROs). See "LRO Manager" on page 1028 for details.
- The Lock Manager handles the locking of data blocks to regulate concurrent data access. See "Lock Manager" on page 1028 for details.
- The Transaction Manager tracks transactions and handles internal commit and abort operations. See "Transaction Manager" on page 1029 for details.

Index Manager

The Index Manager manages the database index and provides a fast way of looking up Analytic Services data blocks. The Index Manager determines which portions of the database index to cache in the index cache, and manages the index cache.

The Index Manager controls five components. The following table describes these components:

Component	Description
Index	The method that Analytic Services uses to locate and retrieve data. The term index also refers to the index file.
Index file	File that Analytic Services uses to store data retrieval information. It resides on disk and contains index pages. Analytic Services names index files incrementally on each disk volume, using the naming convention `ess`*`xxxxx`*`.ind`, where *xxxxx* is a number. The first index file on each disk volume is named `ess00001.ind`.

Component	Description
Index page	A subdivision of an index file that contains index entries that point to data blocks.
Index entry	A pointer to a data block. An index entry exists for every intersection of sparse dimensions.
Index cache	A buffer in memory that holds index pages.

The term *index* refers to all index files for a single database. The index can span multiple volumes, and more than one index file can reside on a single volume. Use the disk volumes setting to specify disk spanning parameters. For information on setting the index cache size, see "Sizing the Index Cache" on page 1129. For information about allocating storage space with the disk volumes setting, see "Specifying Disk Volumes" on page 1038.

Allocation Manager

Allocation Manager, part of the Index Manager, performs these tasks:

- Creation and extension of index and data files on disk
- File open and close operations
- Designation of which volume to use for a new file
- Sequence of volume use

When one of these tasks needs to be performed, the Allocation Manager uses this process to allocate space:

1. It attempts to use free space in an existing file.
2. If enough free space is not available, it attempts to expand an existing file.
3. If enough free space is not available in existing files, it creates a new file on the current volume.
4. If it cannot expand an existing file or create a new file on the specified volume, it attempts to use the next specified volume.
5. If all specified volumes are full, an error message is displayed, and the transaction is aborted.

The Allocation Manager allocates space for index and data files based on the database settings for storage.

➤ To check current values and set new values, use any of the following methods:

Tool	Topic	Location
Administration Services	Setting Database Properties	*Essbase Administration Services Online Help*
MaxL	**alter database**	*Technical Reference*
ESSCMD	SETDBSTATEITEM 23	*Technical Reference*

See "Specifying Disk Volumes" on page 1038 for a detailed discussion of how the disk volumes setting works.

For a comprehensive discussion of how Analytic Services stores data, see "Storage Allocation" on page 1037.

Data Block Manager

The Data Block Manager brings data blocks into memory, writes them out to data files, handles data compression, and writes data files to disk. The Data Block Manager controls four components. The following table describes each component:

Component	Description
Data file	A file that contains data blocks. Analytic Services generates the data file upon data load and stores it on disk. Analytic Services names data files incrementally—`ess`*`xxxxx`*`.pag`, where *xxxxx* is a number starting with 00001.
Data block	The primary storage unit within Analytic Services. A data block is a multidimensional array that represents cells of the dense dimensions for a given intersection of sparse dimensions.
Data cache	A buffer in memory that holds uncompressed data blocks.
Data file cache	A buffer in memory that holds compressed data files (`.PAG`).

The size of the data file cache determines how much of the data within the data files can fit into memory at one time. The data cache size and the data block size determine how many data blocks can fit into memory at one time. Data files for a single database can span multiple volumes; more than one database can reside on the same volume. For information on setting the data file cache size and data cache size, see "Sizing the Data File Cache" on page 1130 and "Sizing the Data Cache" on page 1132. For information about allocating storage space with the disk volumes setting, see "Specifying Disk Volumes" on page 1038.

LRO Manager

Linked reporting objects (LROs) enable you to associate objects, such as flat files, with data cells. Using the Spreadsheet Add-in, users can create and store LRO files, with an `.lro` extension.

LRO files are stored in the database directory (`\ARBORPATH\appname\dbname`, for example, `\Essbase\Sample\Basic`).

Analytic Services stores information about linked reporting objects in an LRO catalog. Each catalog resides in its own Analytic Services index page and coexists in an index file with other, non-LRO Analytic Services index pages.

For a comprehensive discussion of linked reporting objects, see Chapter 11, "Linking Objects to Analytic Services Data" and the *Essbase Spreadsheet Add-in User's Guide*.

Lock Manager

The Lock Manager issues locks on data blocks, which in turn controls concurrent access to data.

The *committed access* and *uncommitted access* isolation levels use different locking schemes. For more information on isolation levels and locking, see Chapter 46, "Ensuring Data Integrity."

Transaction Manager

The Transaction Manager controls transactions and commit operations and manages database recovery.

Analytic Services commits data automatically. Commits are triggered by transactions that modify data—data loading, calculating, restructuring, and spreadsheet lock&send operations.

How Analytic Services commits data depends upon whether the transaction isolation level is set to committed or uncommitted access (the default). For detailed explanations of the two isolation levels, see "Committed Access" on page 1056 and "Uncommitted Access" on page 1060.

The Transaction Manager maintains a transaction control table, *database_name*.tct, to track transactions.

For information about commit operations and recovery, see "Recovering from a Crashed Database" on page 1069.

Understanding the Kernel Startup

This list is the sequence of events during an kernel start-up:

1. After the Analytic Server starts, a user connects to it from a client.
2. The user starts a database.
3. Analytic Services loads the database.
4. The Analytic Services Agent passes database settings to the server.
5. The kernel begins its initialization process.
6. The kernel starts its components—the Index Manager, Lock Manager, LRO Manager, Data Block Manager, and Transaction Manager.

If it encounters an error during start up, the Analytic Services Kernel shuts itself down.

Understanding the Precedence of Database Settings

Analytic Services provides default values for some database storage settings in the `essbase.cfg` file. You can leave the default settings, or change their values in two places:

- You can define storage settings for all databases on the Analytic Server by changing values in the `essbase.cfg` file.
- You can define storage settings for a single database by using any of the methods specified in "Specifying and Changing Database Settings" on page 1033.

Changes made for an individual database permanently override `essbase.cfg` settings and Analytic Services defaults for the relevant database until they are changed or withdrawn.

If you change settings at the database level, the changes become effective at different times, as shown in Table 56:

Table 56: Database-level storage settings effectiveness

Setting	When setting becomes effective
• Index cache • Data file cache • Data cache • Cache memory locking • Disk volume	After you stop and restart a database.
Isolation level parameters Concurrency parameters	The first time after setting these values that there are no active transactions.
All other settings	Immediately.

Note: The size of index pages is fixed at 8 K to reduce input-output overhead, as well as to simplify database migration.

Understanding How Essbase Reads Settings

Analytic Services reads the `essbase.cfg` file when you start Analytic Server, and then applies settings to the appropriate databases that you have created using any of the methods described in "Specifying and Changing Database Settings" on page 1033.

Database settings that you specify using Administration Services or ESSCMD/MaxL always override `essbase.cfg` file settings, even if you change a setting in the `essbase.cfg` file after you have applied a setting for a particular database. Only removing a setting triggers Analytic Services to use the `essbase.cfg` file, and then only after restarting Analytic Server.

Viewing Most Recently Entered Settings

➤ To view the most recently entered settings, use any of the following methods:

Tool	Topic	Location
Administration Services	Setting Database Properties	*Essbase Administration Services Online Help*
MaxL	**display database**	*Technical Reference*
ESSCMD	GETDBSTATE GETDBINFO	*Technical Reference*

For information on stopping and starting servers, applications, and databases, see "Starting and Stopping" on page 916.

Customizing Database Settings

You can customize different settings for each database on Analytic Server. The information in this section helps you understand what each setting controls, how to specify settings, and lists examples. For a table of performance-related settings, see Chapter 50, "Improving Analytic Services Performance."

Note: Configure settings that are applied to an entire Analytic Server with the `essbase.cfg` file. For information about how to create this file and about what settings are available, see the *Technical Reference*.

You can customize these major database settings:

Table 57: Major Kernel Settings

Setting	Where to Find More Information
Index cache size	"Sizing the Index Cache" on page 1129
Data file cache size	"Sizing the Data File Cache" on page 1130
Data cache size	"Sizing the Data Cache" on page 1132
Cache memory locking	"Deciding Whether to Use Cache Memory Locking" on page 1127
Disk volumes	"Storage Allocation" on page 1037
Data compression	"Data Compression" on page 1044
Isolation level	"Understanding Isolation Levels" on page 1054

The following sections describe how to change kernel settings and list examples:

- "Specifying and Changing Database Settings" on page 1033
- "Using alter database in MaxL" on page 1034
- "Using SETDBSTATEITEM in ESSCMD" on page 1034

Specifying and Changing Database Settings

Before you change any database settings, be sure to review information about precedence of the settings as changed in different parts of Analytic Services, and how those settings are read by Analytic Services:

- "Understanding the Kernel Startup" on page 1029
- "Understanding How Essbase Reads Settings" on page 1031

➤ To specify most database settings, use any of the following methods:

Tool	Topic	Location
Administration Services	Setting Database Properties	*Essbase Administration Services Online Help*
MaxL	**alter database**	*Technical Reference*
ESSCMD	SETDBSTATEITEM SETDBSTATE	*Technical Reference*

These different methods provide different ways to change the same database settings. In rare cases, you may want to use the `essbase.cfg` file to specify settings.

CAUTION: In previous versions of Analytic Services, you can specify many database settings in the `essbase.cfg` file on Analytic Server. In Version 5.*x* and higher, Analytic Services overrides most of the `.cfg` settings. For an explanation of how newer versions of Analytic Services handle settings, see "Understanding the Precedence of Database Settings" on page 1030 and "Understanding How Essbase Reads Settings" on page 1031.

Using *alter database* in MaxL

Issue a separate **alter database** statement for each database setting you want to change. For example, the following MaxL script logs on to Analytic Services, changes three database settings, and logs off:

```
login admin identified by secretword;
alter database sample.basic enable committed_mode;
alter database sample.basic set lock_timeout immediate;
alter database sample.basic disable create_blocks;
logout;
```

Note: Terminate each MaxL statement with a semicolon when issuing them using the MaxL Command Shell; however, if MaxL statements are embedded in Perl scripts, do *not* use the semicolon statement terminator.

You can use MaxL to write batch scripts that automate database setting changes. For detailed explanations of MaxL statements, see the *MaxL Language Reference*, located in the *Technical Reference*.

Using SETDBSTATEITEM in ESSCMD

For simple items, specify the command, item number representing the parameter, application, database, and value for the parameter:

```
SETDBSTATEITEM 2 "SAMPLE" "BASIC" "Y";
```

For parameters that require multiple values, such as Isolation Level (item 18), specify multiple values, in this case, all the values after “BASIC”:

```
SETDBSTATEITEM 18 "SAMPLE" "BASIC" "1" "Y" "-1";
```

If you do not know the parameter number, omit it, and Analytic Services lists all parameters and their corresponding numbers. Analytic Services also prompts you for a database and an application name.

Use a separate SETDBSTATEITEM command for each parameter; you cannot string parameter numbers together on the same line.

See the *Technical Reference* for information about the parameters for the SETDBSTATE and SETDBSTATEITEM commands.

Note: SETDBSTATEITEM or SETDBSTATE affect only the specified database.

You can include SETDBSTATEITEM (or SETDBSTATE) in batch scripts. For a comprehensive discussion of batch processing, see "Using Script and Batch Files for Batch Processing" on page 1403. For information on specific ESSCMD syntax, see the *Technical Reference*.

Chapter

45 Allocating Storage and Compressing Data

This chapter explains how Analytic Services stores and compresses data on disk.

This chapter includes the following sections:

Storage Allocation

Analytic Services uses a data file to store data blocks. By default, a data file is located in its associated database folder. Data files follow the naming convention `essn.pag`, where *n* is greater than or equal to one and less than or equal to 65,535.

Analytic Services uses an index file to store the index for a database. By default, an index file is located in its associated database folder. Index files follow the naming convention `essn.ind`, where *n* is greater than or equal to one and less than or equal to 65,535.

Analytic Services automatically allocates storage for data and index files. You can use disk volumes to control how storage is allocated for these files.

➤ To specify disk volumes so that you control how storage is allocated, use this procedure:

1. Verify how much space Analytic Services uses to store index and data files. See "Checking Index and Data File Sizes" on page 1038 for information about how to check sizes.
2. Choose a technique to control storage:
   - Specify which volumes (drives) Analytic Services uses to store these files. See "Specifying Disk Volumes" on page 1038 for a comprehensive discussion of using disk volumes to specify where Analytic Services stores index files.
   - Install Analytic Services on one volume and store files on another volume. See *Essbase Analytic Services Installation Guide* for instructions.

Checking Index and Data File Sizes

➤ To view index file (`.ind` file) and data file (`.pag` file) names, counts, sizes, and totals, and to determine whether each file is presently open in Analytic Services, use either of the following methods:

Tool	Topic	Location
Administration Services	Checking Index and Data File Sizes	*Essbase Administration Services Online Help*
ESSCMD	LISTFILES	*Technical Reference*

Note: The file size information that is provided by the Windows NT operating system for index and data files that reside on NTFS volumes may not be accurate. The file size information provided by Administration Services and by LISTFILES is accurate.

Specifying Disk Volumes

Use disk volumes to specify where you want to store Analytic Services index files (`ess`*n*`.ind`) and data files (`ess`*n*`.pag`). If you do not use the disk volumes setting, Analytic Services stores data only on the volume where the *ARBORPATH*

directory resides. If the *ARBORPATH* variable is not set, Analytic Services stores data only on the volume where the server was started. For instructions on setting *ARBORPATH*, see the *Essbase Analytic Services Installation Guide*.

Note: For information about how to check the size of the index and data files, see "Checking Index and Data File Sizes" on page 1038.

You can specify disk volumes using Administration Services, MaxL, or ESSCMD. When you use disk volumes, Analytic Services provides the following options for each volume:

- Name of the volume
- Maximum space to use on the volume (called Partition Size in Administration Services and Volume Size in ESSCMD).
- File type. You can specify index files, data files, or both. The default is index and data files on the same volume.
- Maximum file size. The default and recommended value is 2,097,152 kilobytes (two gigabytes). When Analytic Services reaches the maximum file size, it creates a new file and names it incrementally. For example, when `ess00001.ind` is filled to maximum size, Analytic Services creates `ess00002.ind`.

CAUTION: If you specify a volume name but not a volume size, Analytic Services uses all available space on the volume.

Analytic Services creates new data files and index files in either of these situations:

- If the total sizes of all files reach the maximum size you have specified in the disk volumes setting. By default, the total is the sum of all index and data file sizes. If you specify Index as the file type, the total refers to the sum of all index files on a particular volume. Likewise, if you specify Data as the file type, the total refers to the sum of all data files on a particular volume.

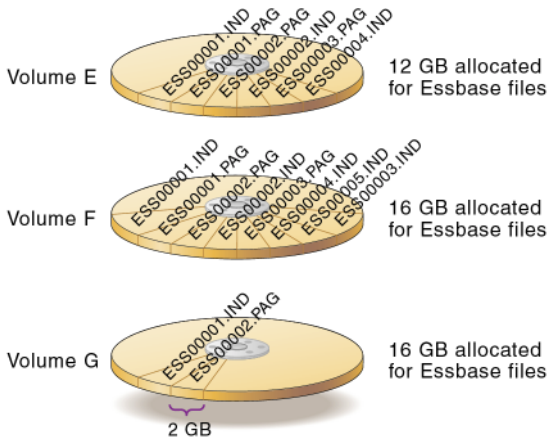
  For example, suppose you want to use up to 12 GB for Analytic Services files on volume E, 16 GB on volume F, and 16 GB on volume G. Analytic Services creates a new file on volume F when the sizes of the index and data files reach 12 GB on volume E and more data needs to be written out to disk.

- If the size of an individual index or data file on any volume reaches a size of 2 GB. In the above example, suppose volumes E and F have reached their capacities and Analytic Services is using volume G. Figure 230 illustrates this example.

  On volume G, Analytic Services creates file `ess00001.ind` and fills it to the default limit of 2 GB. On volume G, Analytic Services creates file `ess00001.pag` and fills it to 1 GB.

  You have specified a limit of 16 GB on volume G, and you have used 3 GB. You have 13 GB left to use on volume g, but `ess00001.ind` has reached the maximum file size of 2 GB. The next time Analytic Services needs storage space when writing index files to disk, Analytic Services creates a new file on volume G and names it `ess00002.ind`. Analytic Services then fills `ess00002.ind` to its 2 GB limit and creates `ess00003.ind`. Analytic Services follows the same procedures for data files.

  Figure 230: Example of How Analytic Services Stores Files Across Volumes



Analytic Services names files consecutively, starting with `ess00001.xxx`, where *xxx* is `ind` for an index file and `PAG` for a data file, and continuing up to `ess65535.xxx`. This naming convention applies to each volume, so in the above example, volumes E, F, and G each have files named `ess00001.pag` and `ess00001.ind`.

Keep in mind the following guidelines when specifying disk volumes:

- Specify the disk volumes in the order in which you want the volumes to be used. You do not have to specify the volume on which Analytic Services is installed as one of the volumes; you can install on one volume and store data on other volumes.
- If a volume reaches capacity, Analytic Services moves on to the next volume.
- If all specified volumes reach capacity, Analytic Services stops any ongoing database operations, issues an error message, and performs fatal error handling. For more information, see "Understanding Fatal Error Handling" on page 1351. If these events occur, shut down the database, allocate more disk space, and restart the database.
- You can tell Analytic Services to stop storing files on a particular volume. Analytic Services can still access the volume as needed, but it no longer stores additional index and data information on the volume. To stop storing information on a volume, select the volume definition that you want to remove and click Delete.
- You set disk volumes on a per database basis. It is possible for more than one database to use space on the same volume, so allocate space carefully. For example, if you specify 7 GB on Volume A for Database 1 and 7 GB on Volume A for Database 2, you have allocated 14 GB for Analytic Services files on Volume A.
- For new files, changes to the disk volumes setting take effect when you next start the database. Existing files and volumes are not affected.

Specifying Disk Volumes with Administration Services

➤ To specify disk volumes with Administration Services, see "Setting Disk Volumes" in *Essbase Administration Services Online Help*.

Specifying Disk Volumes with ESSCMD

➤ To allocate a new volume, enter `SETDBSTATEITEM 23` in ESSCMD and either follow the prompts or supply the required values on the command line.

ESSCMD prompts you for the number of new disk volumes you want to add, unless you supply the number on the command line.

Then, for each new volume, ESSCMD prompts you for the following values, unless you supply them on the command line.

- Volume name (for each volume).
- Volume size (maximum space to use on the volume)—The default value is Unlimited; the minimum setting is 8 megabytes.

  When you use ESSCMD, you can specify volume size in bytes (B), kilobytes (K), megabytes (M), gigabytes (G), or terabytes (T). ESSCMD displays minimum, maximum, and current values and 0 for unlimited.
- File type—You can specify index files, data files, or both. The default is 3-Index + Data (index and data files on the same volume).
- File size (maximum size that each file specified in file type can attain before Analytic Services creates a new file)—The default value is 2 gigabytes; the minimum setting is 8 megabytes.

  When you use ESSCMD, you can specify file size in bytes (B), kilobytes (K), megabytes (M), or gigabytes (G). ESSCMD displays minimum, maximum, and current values.

The following example allocates up to 10 gigabytes on Volume E, sets a maximum file size of 2 gigabytes, and specifies that data files should be stored only on E:

```
SETDBSTATEITEM 23 "SAMPLE" "BASIC" "1" "E" "10G" "2" "2G"
```

➤ To change the settings on an allocated volume, enter SETDBSTATEITEM 24 in ESSCMD and either follow the prompts or supply the required values on the command line.

ESSCMD prompts you for the following values, unless you supply them on the command line:

- Volume number. (Use the GETDBSTATE command in ESSCMD to see a list of the currently defined disk volumes and to see the number assigned to each volume.)
- Volume name
- Volume size
- File type
- File size

The following example allocates up to 20 gigabytes on Volume C and sets a maximum file size of 2 gigabytes:

```
SETDBSTATEITEM 24 "SAMPLE" "BASIC" "1" "C" "20G" "3" "2G"
```

- To stop Analytic Services from storing additional files on a volume, enter SETDBSTATEITEM 25 in ESSCMD and either follow the prompts or supply the required values on the command line. Analytic Services continues accessing files on the deallocated volume, but does not write new files to it.

  ESSCMD prompts you for the following value, unless you supply it on the command line—Delete which volume definition. Use the GETDBSTATE command in ESSCMD to see a list of the currently defined disk volumes and to see the number assigned to each volume.

  The following example deallocates the volume that is specified as fourth:

  ```
  SETDBSTATEITEM 25 "SAMPLE" "BASIC" "4"
  ```

  Note: If you delete an application or database, Analytic Services does not remove the directory containing the application or database on a disk volume. The computer's operating system still shows the folder and file labels on the disk. However, you can reuse the same name of the application or database that you had removed on the disk volume.

  For more syntax information, see the *Technical Reference*.

  On UNIX, *volume_name* is a mounted UNIX file system. You must enter a fully qualified path name up to the name of the directory you are using for Analytic Services. Analytic Services automatically appends the `\app` directory to the path; you do not specify the `\app` directory.

  Consider the following examples:

  ```
  /vol2/essbase 10M
  ```

  Volume size is the maximum space, in kilobytes, allocated to the volume. The default value is unlimited—Analytic Services uses all available space on that volume.

Reviewing an Example of Specifying Volumes To Control Storage

Assume you want to use up to 20 GB for Analytic Services files on Volume E, 25 GB on Volume F, and 25 GB on Volume G. You are using the default file size limit of 2 GB.

When you load data, Analytic Services stores up to 20 GB on Volume E; if the database is larger than 20 GB, Analytic Services stores the next 25 GB on Volume F, and so on.

Figure 231: Example of Using Disk Volumes

Disk Volume	Partition Size	File Type	File Size
E	20971520 K	Index+Data	2097152 K
F	26214400 K	Index+Data	2097152 K
G	26214400 K	Index+Data	2097152 K

Data Compression

Analytic Services allows you to choose whether or not data blocks that are stored on disk are compressed, as well as which compression scheme to use. When data compression is enabled, Analytic Services compresses data blocks when it writes them out to disk. Analytic Services fully expands the compressed data blocks, including empty cells, when the blocks are swapped into the data cache.

Generally, data compression optimizes storage use. You can check compression efficiency by checking the compression ratio statistic. See “Checking the Compression Ratio” on page 1051 for a review of methods.

Analytic Services provides several options for data compression:

- Bitmap compression, the default. Analytic Services stores only non-missing values and uses a bitmapping scheme.
- Run-length encoding (RLE). Analytic Services compresses repetitive, consecutive values, including zeros and #MISSING values.
- zlib compression. Analytic Services builds a data dictionary based on the actual data being compressed.
- Index Value Pair compression. Analytic Services applies this compression if the block density is less than 3%.
- No compression. Analytic Services does not compress data blocks when they are written to disk.

Because Analytic Services compresses data blocks as they are written to disk, it is possible for bitmap, RLE, and uncompressed data blocks to coexist in the same data file. Keep in mind the following rules:

- When a compressed data block is brought into the data cache, Analytic Services expands the block to its full size, regardless of the scheme that was used to compress it.
- When Analytic Services stores a block on disk, Analytic Services treats the block the same whether the block was compressed or uncompressed when it was brought into the data cache. In either case, Analytic Services compresses the block according to the specified compression type (including not compressing the block if no compression is specified).
- If compression is not enabled, Analytic Services writes out the fully expanded block to disk.

You may want to disable data compression if blocks have very high density (90% or greater) and have few consecutive, repeating data values. Under these conditions, enabling compression consumes resources unnecessarily.

Bitmap Data Compression

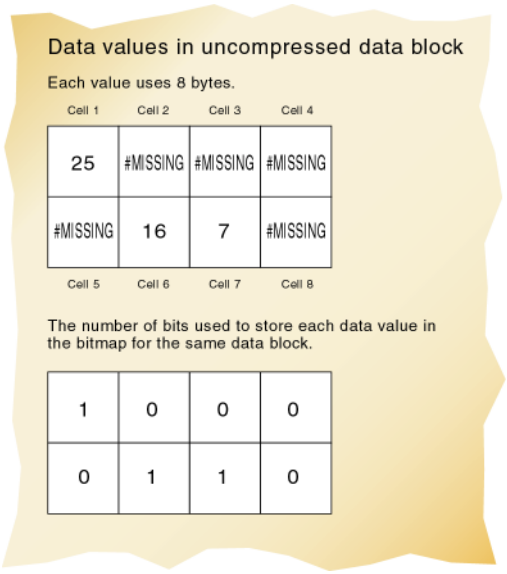
With bitmap compression, Analytic Services uses a bitmap to represent data cells, and stores only the bitmap, the block header, and the other control information. A bitmap uses one bit for each cell in the data block, whether the cell value is missing or non-missing. When a data block is not compressed, Analytic Services uses 8 bytes to store every non-missing cell.

When using bitmap compression, Analytic Services stores only non-missing values and does not compress repetitive values or zeros (contrast with RLE compression, described in "RLE Data Compression" on page 1047). When Analytic Services pages a data block into the data cache, it fully expands the data block, using the bitmap to recreate the missing values.

Because the bitmap uses one bit for each cell in the data block, the bitmap scheme provides a fixed overhead for data compression. Figure 232 represents a portion of a data block, as an example. In this example, Analytic Services uses 64 bytes to

store the data in the fully expanded block, but uses one byte (eight bits) to store the bitmap of the compressed data on disk. (Analytic Services also uses a 72-byte block header for each block, whether the block is compressed or not.)

Figure 232: Bitmap Data Compression



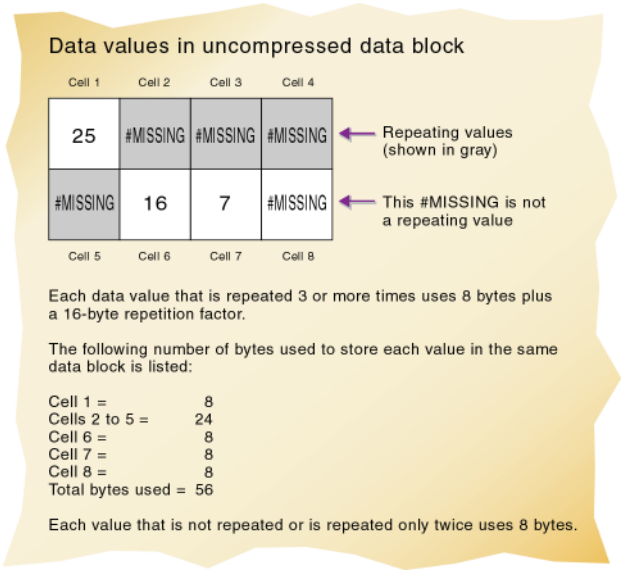
In most cases, bitmap compression conserves disk space more efficiently. However, much depends on the configuration of the data.

RLE Data Compression

When using the run-length encoding (RLE) compression scheme, Analytic Services compresses any consecutive, repetitive values—any value that repeats three or more times consecutively, including zero. Analytic Services keeps track of each repeating value and the number of times it is repeated consecutively.

In the example in Figure 233, Analytic Services uses 64 bytes to store the data in the fully expanded block, but uses 56 bytes to store the compressed data on disk. (Analytic Services also uses a 72-byte block header for each block, whether the block is compressed or not.)

Figure 233: RLE Data Compression



zlib Compression

This method is used in packages like PNG, Zip, and gzip. Calculation and data loading is faster with direct I/O and zlib compression than with buffered I/O and zlib compression. If data storage is your greatest limiting factor, use zlib, but be aware that, under some circumstances, data loads may be up to 10% slower than bitmap compression. On the other hand, the size of the database is generally significantly smaller when you use zlib compression.

In contrast to bitmap compression, which uses an algorithm to track which values are missing and does not interact with any other type of data, zlib compression builds a data dictionary based on the actual data being compressed (including any missing values). Therefore, zlib compression should provide greater compression ratios over bitmap compression given extremely dense data. However, because the effectiveness of the zlib algorithm is dependent (at the bit level) on the actual data being compressed, general guidelines about when zlib compression provides greater compression than bitmap compression based solely on density are not available. Unlike other compression methods, the storage space saved has little or no relationship to the number of missing cells or the number of contiguous cells of equal value. Generally, the more dense or heterogeneous the data is, the better zlib will compress it in comparison to bitmap or RLE compression. However, under some circumstances, it is possible that zlib will not yield better results than using bitmap or RLE compression. It is best to test with a representative sample of data.

To estimate the storage savings you may obtain with zlib, create a small database using your normal compression technique (bitmap or RLE) with a small sampling of real data and shut down Analytic Server. Note the size of the created data files. Then clear the data in the sample database, change the compression setting to zlib, reload the same sample data, and shut down Analytic Server again. Now note the difference in the storage used. You can also use the small sample database to estimate any changes in calculation or data loading speed.

Index Value Pair Compression

Index Value Pair addresses compression on databases with larger block sizes, where the blocks are highly sparse. This compression algorithm is not selectable, but is automatically used whenever appropriate by the database. The user must still choose between the compression types None, bitmap, RLE, and zlib through Administration Services.

For example, if the user selects RLE, Analytic Services reviews each block and evaluates the following compression types for highest compression: RLE, bitmap, or Index Value Pair. On the other hand, if the user chooses zlib, for example, zlib is the only compression type applied.

The following table illustrates the available compression types the user can choose and the compression types that Analytic Services evaluates, then applies.

Chosen Compression Type	Evaluated Compression Type
None	None
Bitmap	Bitmap, Index Value Pair
RLE	RLE, Bitmap, Index Value Pair
zlib	zlib

Deciding Which Type of Compression to Use

You can choose from four compression settings: bitmap (the default), RLE, zlib, or None.

In most cases, you need not worry about choosing a compression setting. Bitmap compression almost always provides the best combination of fast performance and small data files. However, much depends on the configuration of the data.

Data compression is CPU-intensive. You need to consider the trade-offs of computation costs versus I/O costs and disk space costs when determining which compression setting to use.

In general, a database compresses better using the RLE setting than the bitmap setting if a large number of repeated non-missing data cells for a given block have the same value. Using RLE compression is computationally more expensive than using bitmap compression. Although if your database shrinks significantly using RLE compression, you may actually see a performance improvement due to decreased I/O costs.

Most databases shrink when using zlib compression, but this is not always the case. Using zlib compression significantly increases the CPU processing. For most databases, this extra CPU processing outweighs the benefits of the decreased block size. But, if your database shrinks significantly using zlib compression, you may see a performance improvement due to decreased I/O costs.

The None compression setting does not reduce the disk usage of a database compared to bitmap compression. In fact, no compression may make no difference to improve the performance of the database because bitmap compression is extremely fast.

Remember that each database is unique and the previous statements are general characteristics of each compression type of which you can choose. Although the default bitmap compression works well for almost every database, the most reliable way to determine which compression setting is best for your database is to try out each one.

Changing Data Compression Settings

Changes to the data compression setting take effect immediately as Analytic Services writes data blocks to disk. For blocks already on disk, Analytic Services does not change compression schemes or enable or disable compression. When you change the data compression settings of blocks already on disk, Analytic Services uses the new compression scheme the next time Analytic Services accesses, updates, and stores the blocks.

➤ To view or change the current settings, use any of the following methods:

Tool	Topic	Location
Administration Services	Selecting a Data Compression Method	*Essbase Administration Services Online Help*
MaxL	**alter database**	*Technical Reference*
ESSCMD	To enable or disable data compression: SETDBSTATE or: SETDBSTATEITEM 14 To set the data compression type: SETDBSTATEITEM 15	*Technical Reference*

Example of Using SETDBSTATEITEM

➤ To enable or disable data compression, enter SETDBSTATEITEM 14 in ESSCMD and either follow the prompts or supply the required values on the command line.

ESSCMD prompts you for the following values, unless you supply them on the command line:

- Data Compression on Disk? Enter Y (Yes, the default) or N (No).
- Data Compression Type. Enter 1 (run-length encoding) or 2 (bitmap, the default).

➤ To specify the data compression type, enter SETDBSTATEITEM 15 in ESSCMD and either follow the prompts or supply the required values on the command line. ESSCMD prompts you for a value of "1" (run length encoding) or "2" (bitmap, the default).

The following example enables Bitmap compression:

```
SETDBSTATEITEM 14 "SAMPLE" "BASIC" "Y" "2"
```

For more syntax information, see the *Technical Reference*.

Checking the Compression Ratio

The compression ratio represents the ratio of the compressed block size (including overhead) to the uncompressed block size, regardless of the compression type in effect. Overhead is the space required by mechanisms that manage compression/expansion.

➤ To check the compression ratio, use either of the following methods:

Tool	Topic	Location
Administration Services	Checking the Compression Ratio	*Essbase Administration Services Online Help*
ESSCMD	GETDBSTATS	*Technical Reference*

Note: The larger the number, the more compression. The compression ratio can vary widely from block to block.

Data Block Size

Data block size is determined by the amount of data in a particular combination of dense dimensions. For example, when you change the dense or sparse configuration of one or more dimensions in the database, the data block size changes. Data block size is 8*n* bytes, where *n* is the number of cells that exist for that combination of dense dimensions.

Note: Eight to 100 kilobytes is the optimum size range.

For information about and instructions for determining the size of a data block, see "Size of Expanded Data Block" on page 1361.

➤ To view the block size for a database, use either of the following methods:

Tool	Topic	Location
Administration Services	Checking Data Block Statistics	*Essbase Administration Services Online Help*
ESSCMD	GETDBSTATS	*Technical Reference*

Chapter

46

Ensuring Data Integrity

This chapter describes how Analytic Services handles transactions and locking and other ways that Analytic Services protects data.

This chapter includes the following sections:

The information in this chapter is not relevant to aggregate storage databases.

Understanding Transactions

When a database is in read/write mode, Analytic Services considers every update request to the server (such as a data load, a calculation, or a statement in a calculation script) as a transaction. Analytic Services tracks information about transactions in a transaction control file (*dbname*.tct).

The transaction control file contains an entry for each transaction and tracks the current state of each transaction (Active, Committed, or Aborted).

For a detailed description of the transaction process, see "Understanding How Analytic Services Handles Transactions" on page 1064.

Understanding Isolation Levels

Isolation levels determine how Analytic Services commits data to disk. When data is committed, it is taken from server memory and written to the database on disk. Analytic Services automatically commits data to disk. There are no explicit commands that users perform to commit data blocks. However, setting the isolation level for a database defines how Analytic Services automatically commits data blocks.

Analytic Services offers two isolation levels for transactions—*committed access* and *uncommitted access* (the default). You can optimize data integrity by using committed access.

For an explanation of each type of access, see "Committed Access" on page 1056 and "Uncommitted Access" on page 1060.

Note: The Spreadsheet Add-in lock and Send and the Grid API are always in Committed Access Mode.

Data Locks

Analytic Services issues write (exclusive) locks for blocks that are created, updated, or deleted, and issues read (shared) locks for blocks that need to be accessed but not modified. By issuing the appropriate locks, Analytic Services ensures that data changed by one operation cannot be corrupted by a concurrent update.

This section discusses locks on data blocks, not locks on database objects. For information about locking and unlocking outlines and other objects, see "Locking and Unlocking Objects" on page 964.

This table explains the lock types:

Table 58: Basic Lock Types

Lock	Description
Write (exclusive) lock	Prevents any other transaction from accessing the locked data block. Used for all data block updates, including spreadsheet lock&send.
Read (shared) lock	Allows other transactions read-only access to the data block, but prevents other transactions from modifying the data block.

This table shows the locks that Analytic Services issues for various types of operations.

Table 59: Locking by Higher-Level Functions

Type of Operation	Lock Issued
Spreadsheet retrieve	Read (shared) lock on each data block.
Retrieve and lock	Write (exclusive) lock on all affected blocks. A subsequent send command commits the data.
Calculate derived block	Write lock on the block being calculated. As a block is calculated, all blocks containing the block's children acquire read locks.
Data load	Write lock.
Restructure	Write lock.

How Analytic Services handles locking depends on whether committed or uncommitted access is enabled.

Committed Access

Committed access provides a high level of data consistency because only one transaction at a time can update data blocks. Under committed access, Analytic Services allows transactions to hold read/write locks on all data blocks involved with the transaction until the transaction completes and commits. However, you can still allow read-only access to the last committed data values.

Analytic Services provides options that determine when locks are issued on data blocks:

- Pre-image access (enabled by default). Pre-image access provides users read-only access to data blocks that are locked for the duration of a concurrent transaction. Users see the last committed data values for the locked data blocks.
- Wait, or time out:
  - Indefinite wait (the default). The transaction waits to acquire a lock on the required locked block.
  - Immediate access, or no wait. If a required block is locked by another transaction, Analytic Services displays a lock time out message, and the transaction aborts.
  - A number of seconds that you specify. The transaction waits that number of seconds to acquire a lock on the required locked blocks. If the specified time runs out before the transaction acquires a lock, Analytic Services displays a lock time out message, and the transaction aborts.

When you have pre-image access enabled, you are not limited to read-only access to data blocks; if you need write access to locked blocks, the transaction waits for write access or times out, depending on the wait or time-out setting. The transaction gets immediate write access to data blocks that are not locked by another transaction.

If you do not have pre-image access enabled and if you need read or write access to locked blocks, the transaction waits for write access or times out, depending on the wait or time-out setting.

Memory Considerations with Committed Access

Under committed access, note the following memory considerations:

- Analytic Services retains redundant data until a transaction commits. Allow disk space for double the size of the database to accommodate redundant data.
- Models with a large number of blocks may experience memory problems under committed access. Each lock (one lock per block) uses approximately 80 bytes of memory per calculation, and each lock is held in memory until the transaction is complete. There is a limit to the addressable memory space per process and eventually models with a large number of blocks may hit this limit, causing the transaction to terminate. In such cases, consider using Uncommitted Access.

Locking Under Committed Access

Under committed access, Analytic Services locks blocks for read and write access:

- For read access, the lock remains until another transaction requests it, whether or not the transaction is complete. Other transactions can read the locked block, but none can alter it.
- For write access, a transaction locks and holds the lock on each block that it modifies until the transaction completes.

Table 60 illustrates locking behavior under committed access when more than one transaction is contending for a lock on the same data. In the example in Table 60, transaction Tx1 is running, and transaction Tx2 is requesting access to the same data.

Note that access to locked blocks depends on what options are enabled. For a discussion of options, see "Committed Access" on page 1056.

Table 60: Locking Behavior Under Committed Access

		Tx1 holds read lock; Tx2 requests read lock	**Tx1 holds read lock; Tx2 requests write lock**	**Tx1 holds write lock; Tx2 requests read lock**	**Tx1 holds write lock; Tx2 requests write lock**
Pre-image access enabled	Wait (time-out) period specified (indefinite wait or a number of seconds wait)	Tx2 gets read lock.	Tx2 waits for Tx1 to release read lock.	Tx2 gets pre-image access.	Tx2 waits for Tx1 to release write lock.
	No wait (time-out) period specified (immediate time-out)	Tx2 gets read lock.	Analytic Services issues time-out message and aborts the transaction.	Tx2 gets pre-image access.	Analytic Services issues time-out message and aborts the Tx2 transaction.
No pre-image access	Wait (time-out) period specified (indefinite wait or a number of seconds wait)	Tx2 gets read lock.	Tx2 waits for Tx1 to release read lock.	Tx2 waits for Tx1 to release write lock.	Tx2 waits for Tx1 to release write lock.
	No wait (time-out) period specified (immediate time-out)	Tx2 gets read lock.	Analytic Services issues time-out message and aborts the Tx2 transaction.	Analytic Services issues time-out message and aborts the Tx2 transaction.	Analytic Services issues time-out message and aborts the Tx2 transaction.

For information about how to set concurrency parameters, see "Specifying Data Integrity Settings" on page 1065.

Concurrency with Committed Access

Occasionally under committed access, a situation results when two transactions are locking or waiting for access to the same blocks, and neither transaction can complete under these conditions. This situation called a deadlock.

For example, if transaction Tx1 needs to update first data block B1 and then data block B2, it first locks B1 and then attempts to lock B2. Meanwhile, if transaction Tx2 needs to update first data block B2 and then block B1, Tx2 first locks B2 and then attempts to lock B1. Tx1 locked B1 and is waiting for B2, and Tx2 locked B2 and is waiting for B1.

Analytic Services transactions periodically perform deadlock detection while waiting to acquire a lock. If detected, Analytic Services issues an error message, and the transaction aborts.

If you try to update a block that is locked by another user, Analytic Services behaves as follows:

- If wait is set to indefinite, the transaction waits to acquire the needed locks.
- If wait is set to 0 (immediate) and if the required blocks are not immediately available, Analytic Services displays an error message, and aborts the transaction.
- If wait is set to a user-specified number of seconds and the time has expired, Analytic Services displays an error message and aborts the transaction.
- If the request times out, try the operation again.

For information about how to set concurrency options, see "Specifying Data Integrity Settings" on page 1065.

Rollback with Committed Access

Under committed access, if the server crashes, Analytic Services rolls back all database updates by transactions that were in progress when the server stopped. Thus, Analytic Services ensures that changes made by the aborted transactions are undone.

If a transaction is aborted due to a non-fatal error, all changes made by the transaction are rolled back.

For a description of the recovery process, see "Recovering from a Crashed Database" on page 1069.

Uncommitted Access

With uncommitted access (enabled by default), the Analytic Services kernel allows transactions to hold read/write locks on a block-by-block basis; Analytic Services releases a block after it is updated but does not commit blocks until the transaction completes or until a specified limit (a "synchronization point") has been reached. You can set this limit, as described below.

Concurrent users accessing the same data blocks might experience unexpected results under uncommitted access, because Analytic Services allows read-only access to data at its last commit point.

With uncommitted access, you can control when Analytic Services performs an explicit commit operation by specifying synchronization point parameters:

- Commit Blocks (number of blocks modified before a synchronization point occurs). The default is 3,000.

  If you set Commit Blocks to 0, the synchronization point occurs at the end of the transaction.

- Commit Rows (number of rows to data load before a synchronization point occurs). The default is 0, which means that the synchronization point occurs at the end of the data load.

- If either Commit Blocks or Commit Rows has a non-zero value, a synchronization point occurs when the first threshold is reached. For example, if Commit Blocks is 10 but Commit Rows is 0 and you load data, a synchronization point occurs after 10 blocks are updated. If Commit Blocks is 5 and Commit Rows is 5 and you load data, a synchronization point occurs after 5 rows are loaded, or 5 blocks are updated, whichever happens first.

If a user-defined threshold is exceeded during an operation, Analytic Services issues a synchronization point to commit the data processed to that point. Analytic Services performs as many synchronization points as are necessary to complete the operation.

Note: Analytic Services analyzes the value of Commit Blocks and Commit Rows during its analysis of feasibility for parallel calculation use. If Analytic Services finds the values set too low, it automatically increases them.

For information about how to specify synchronization point parameters, see "Specifying Data Integrity Settings" on page 1065.

CAUTION: Analytic Services retains redundant data to enforce transactional semantics. Allow disk space for double the size of the database to accommodate redundant data, particularly if both Commit Blocks and Commit Rows are set to 0.

Memory Considerations with Uncommitted Access

If your data cache is too small to hold the number of blocks specified in your Commit Blocks and Commit Rows settings, then note that blocks will be written to disk as soon as the caches become full, which will be before the transaction is committed.

Locking Under Uncommitted Access

Under uncommitted access, Analytic Services locks blocks for write access until Analytic Services finishes updating the block. Uncommitted access is in contrast to committed access, when Analytic Services holds locks until a transaction completes.

Table 61 illustrates locking behavior under uncommitted access when more than one transaction contends for a lock on the same data. In the example in Table 61, transaction Tx1 is running, and transaction Tx2 is requesting access to the same data.

Table 61: Locking Behavior with Uncommitted Access

Status When Tx2 Makes a Request	If Tx1 holds read lock	If Tx1 holds write lock
Read lock	Tx2 gets read lock.	Tx2 gets read lock.
Write lock	Tx2 gets write lock.	Tx2 waits for Tx1 to release the lock.

Concurrency with Uncommitted Access

With uncommitted access, blocks are released more frequently than with committed access, when all blocks are locked until the end of the transaction.

Rollback with Uncommitted Access

Under uncommitted access, if the server crashes, Analytic Services rolls back all database updates from the point of the last successful commit. Some of the updates from an aborted transaction may have committed. Whether transactions committed their updates the way users expected depends on the order in which overlapping transactions updated and committed data.

If a transaction is aborted due to a non-fatal error, Analytic Services commits only the data that the transaction finished processing prior to the abort of the transaction.

For a description of the recovery process, see "Recovering from a Crashed Database" on page 1069.

Parallel Calculation and Uncommitted Access

If Analytic Services is using parallel calculation, Analytic Services checks the commit threshold.

Committed Versus Uncommitted Access

Consider these issues when choosing an isolation level:

Table 62: Issues Affecting Selection Of An Isolation Level

Issue	Explanation
Database performance	Uncommitted access always yields better database performance than committed access. When using uncommitted access, Analytic Services does not create locks that are held for the duration of a transaction but commits data based upon short-term write locks.
Data consistency	Committed access provides a higher level of data consistency than uncommitted access. Retrievals from a database are more consistent. Also, only one transaction at a time can update data blocks when the isolation level is set to committed access. This factor is important in databases where multiple transactions attempt to update the database simultaneously.
Data concurrency	Uncommitted access provides better data concurrency than does committed access. Blocks are released more frequently than they are during committed access. With committed access, deadlocks can occur.
Database rollbacks	If a server crash or other server interruption occurs while there are active transactions running, the Analytic Services kernel rolls back the transactions when the server is restarted. With committed access, rollbacks return the database to its state before any transactions began. With uncommitted access, rollbacks may result in some data being committed and some data not being committed. For information about actions to take when a transaction does not complete, see “What to Expect If a Server Interruption Occurs” on page 1071.

Understanding How Analytic Services Handles Transactions

Analytic Services tracks transactions from start to finish, swapping data blocks in and out of memory as needed and committing data blocks when a transaction completes. The following list describes how Analytic Services handles a transaction: all list items apply to both committed and uncommitted access (see "Understanding Isolation Levels" on page 1054).

1. A user or batch program begins an operation.
2. The OLAP engine notifies the Analytic Services kernel that a transaction is to begin.
3. The Analytic Services kernel begins the transaction.
4. The OLAP engine requests data from the Analytic Services kernel.
5. The Analytic Services kernel locates the requested data. It passes the data, and some associated control information, to the OLAP engine. If you are using Spreadsheet Add-in, you see this data displayed on the sheet.
6. If you are using Spreadsheet Add-in, when you modify data, you issue the Send command.
7. The Analytic Services kernel associates the transaction with an entry in its transaction control table.
8. After the operation is complete on the OLAP engine side, the OLAP engine notifies the Analytic Services kernel about the update, and the Analytic Services kernel updates internal data structures accordingly.
9. Steps 4 through 8 repeat as often as necessary to complete the operation.
10. The transaction ends. If Analytic Services encounters an error during transaction processing, it aborts the transaction. If no errors are encountered, Analytic Services commits the transaction. See "Understanding Isolation Levels" on page 1054 for details on the differences in commit behavior under committed and uncommitted access.
11. Analytic Services issues a message to notify the client that the transaction is complete; for example, "TOTAL CALC ELAPSED TIME..."

Under uncommitted access, it is possible to access uncommitted data when multiple transactions are active and are accessing the same data. Transaction results are unpredictable under uncommitted access.

Under uncommitted access, if you have defined a commit threshold, Analytic Services may need to break down a single database operation into multiple synchronization points. See "Uncommitted Access" on page 1060 for information on commit thresholds.

Specifying Data Integrity Settings

You can specify isolation level, synchronization point parameters, and concurrency parameters using Administration Services, MaxL, or ESSCMD. Changes to isolation level settings take effect the next time there are no active transactions. For information about deciding which settings to choose, see "Committed Access" on page 1056 and "Uncommitted Access" on page 1060.

➤ To specify data integrity settings, use any of the following methods:

Tool	Topic	Location
Administration Services	Setting Data Integrity Options	*Essbase Administration Services Online Help*
MaxL	**alter database**	*Technical Reference*
ESSCMD	SETDBSTATEITEM 18	*Technical Reference*

Example of Specifying Isolation Level Settings with ESSCMD

➤ To specify isolation level settings using ESSCMD, enter SETDBSTATEITEM 18 in ESSCMD and either follow the prompts or supply the required values on the command line.

Choose 1 (committed access) or 2 (uncommitted access, the default). Depending on the type of access that you specify, ESSCMD prompts you for other parameters (or you can supply the values on the command line).

If you choose 1 (committed access), ESSCMD prompts for the following information:

- Pre-image access; Y (Yes) or N (No, the default). Pre-image access provides users read-only access to data blocks that are locked for the duration of a transaction. Users see the last committed data values for the locked data blocks.
- Wait (in the Database Settings dialog box) or time out (in ESSCMD): -1, 0, or *n*.
  - -1 is indefinite wait.
  - 0 is immediate access, or no wait.
  - *n* is a number of seconds that you specify.

If you choose 2 (uncommitted access), ESSCMD prompts for the following values. See "Uncommitted Access" on page 1060 for explanations of these options.

- Number of blocks modified before internal commit
- Number of rows to data load before internal commit

You can also specify isolation level parameters (pre-image access and so on) by specifying parameters 19–22 on SETDBSTATEITEM. Enter SETDBSTATEITEM with no parameters; ESSCMD displays a list that includes each parameter by number, with a description.

Here is an example of using SETDBSTATEITEM to set an isolation level. This example enables committed access and pre-image access and specifies indefinite wait time.

```
SETDBSTATEITEM 18 "SAMPLE" "BASIC" "1" "Y" "-1"
```

For more syntax information, see the *Technical Reference*.

Example of Specifying Isolation Level Settings with MaxL

To specify isolation level settings using MaxL, use this MaxL statement:

```
alter database dbname enable committed_mode
```

For detailed information about the use of this statement, see the *Technical Reference*.

Accommodating Data Redundancy

To ensure data integrity, the Analytic Services kernel temporarily retains redundant (duplicate) information. To accommodate redundant information, allow disk space for double the size of the database.

Analytic Services maintains a file called *dbname*.esm, in which it stores crucial control information.

CAUTION: The *dbname*.tct file, *dbname*.esm file, the index files, and the data files contain information crucial for data recovery. Never alter or delete these files.

Checking Structural and Data Integrity

To validate database integrity and to check for database corruption, use one of the following methods:

- Perform a dense restructure. Because a dense restructure recreates all blocks within a database, this method verifies index nodes and cells for each block.
- Export all levels of data from the database. Exporting an entire database accesses blocks and all data values across the entire database.
- Use the VALIDATE command (ESSCMD) to check structural and data integrity. For an explanation of how VALIDATE works, see "Using VALIDATE to Check Integrity" on page 1067.

If errors occur during any of these checks, restore the database from backups. For an explanation of how to back up and restore a database, see Chapter 47, "Backing Up and Restoring Data."

Using VALIDATE to Check Integrity

The VALIDATE command performs many structural and data integrity checks:

- Verifies the structural integrity of free space information in the index.
- Compares the data block key in the index page with the data block key in the corresponding data block.

- The Essbase index contains an entry for every data block. For every Read operation, VALIDATE automatically compares the index key in the index page with the index key in the corresponding data block and checks other header information in the block. If it encounters a mismatch, VALIDATE displays an error message and continues processing until it checks the entire database.
- Restructures data blocks whose restructure was deferred with incremental restructuring.
- Checks every block in the database to make sure each value is a valid floating point number.
- Verifies the structural integrity of the linked reporting objects (LROs) catalog.

Note: When you issue the VALIDATE command, we recommend placing the database in read-only mode.

As Analytic Services encounters mismatches, it records error messages in the VALIDATE error log. You can specify a file name for error logging; Analytic Services prompts you for this information if you do not provide it. The VALIDATE utility continues running until it has checked the entire database.

You can use the VALIDATE command in ESSCMD to perform these structural integrity checks.

During index free space validation, the VALIDATE command verifies the structural integrity of free space information in the index. If integrity errors exist, Analytic Services records them in the VALIDATE log. The file that you specified on the VALIDATE command holds the error log.

If VALIDATE detects integrity errors regarding the index free space information, the database must be rebuilt. You can rebuild in any one of three ways:

- Restoring the database from a recent system backup.
- Restoring the data by exporting data from the database; creating a new, empty database; and loading the exported data into the new database.
- Restructuring the database.

  For an explanation of restoring a database, see Chapter 47, “Backing Up and Restoring Data.” For a comprehensive discussion of restructuring, see Chapter 52, “Optimizing Database Restructuring.”

Even if you do not use VALIDATE, Analytic Services automatically performs certain validity checking whenever a read operation is performed, to ensure that the index is properly synchronized with the data.

For every read operation, Analytic Services compares the data block key in the index page with the data block key in the corresponding data block and checks other header information in the block.

If Analytic Services encounters a mismatch, it displays an “Invalid block header” error message.

Recovering from a Crashed Database

After a server interruption such as a crash, Analytic Services recovers a database, rolling back all transactions that were active when the interruption occurred. Recovery time depends on the size of the index. The larger the index, the longer it takes to recover the database.

Analytic Services also recovers and consolidates free fragments (unused addressable units in the data blocks). However, free space recovery is the most time consuming aspect of database recovery, so it is delayed by default. You must trigger free space recovery explicitly unless you have changed the default setting. See “Free Space Recovery” on page 1070 for the advantages and disadvantages of delaying free space recovery.

Analytic Services recovers data as soon as the server is started after a server interruption. Recovery consists of the following phases:

1. Transaction recovery rolls back all transactions that were active when the interruption occurred.
2. Index file recovery truncates files to their previously committed sizes.
3. Data free space recovery rebuilds the data free space tables. The size of the index determines the duration of this phase.

Note: Free space recovery is delayed until you trigger it, unless you have changed the default setting. See “Free Space Recovery” on page 1070.

A media failure (faulty disk, disk failure, or head crash) requires you to restore data from backups. For an explanation of how to backup and restore a database, see Chapter 47, "Backing Up and Restoring Data."

CAUTION: Do not move, copy, modify, or delete any of the following files—essxxxxx.ind, essxxxxx.pag, dbname.ind, dbname.esm, dbname.tct. Doing so can result in data corruption.

The Analytic Services kernel uses fatal error handling to display appropriate messages and to shut down the server, depending on the error encountered. For an explanation of how fatal error handling works, see "Understanding Fatal Error Handling" on page 1351.

For information about how transactions are rolled back after a crash, see "Committed Versus Uncommitted Access" on page 1063.

Free Space Recovery

Database recovery takes place any time you load an application that has just crashed or terminated abnormally. Analytic Services does not perform free space recovery automatically because it is the most expensive part of database recovery. You must either trigger free space recovery explicitly or change the default setting so that Analytic Services will recover free space automatically.

All database functions run normally whether you recover free space or not. The advantage of recovering free space is that you can reuse disk space in the data files that is marked as free. The disadvantage is that free space recovery is time consuming, so you might want to delay recovery until an advantageous time.

You should, however, perform free space recovery as soon as possible to take full advantage of the free space in the data files and to ensure that the database hasn't been corrupted. Also, if a database crashes repeatedly and you do not run free space recovery, the data files can become unnecessarily large.

To trigger free space recovery, use the MaxL alter database command, as follows:

```
alter database DBS-NAME recover freespace
```

See the *Technical Reference* for more information about this MaxL command.

To change the default behavior for free space recovery, change the DELAYEDRECOVERY configuration setting to FALSE. See the "Configuration Settings" section of the *Technical Reference* for more information about this setting.

To get information about free space recovery, use the GETDBSTATS command. GETDBSTATS provides the following information about free space recovery:

```
Free Space is Recoverable                 : true/false
Estimated Bytes of Recoverable Free Space : nnn
```

Note: If free space is recoverable, the block counters are estimates and do not necessarily match the number of existing blocks.

What to Expect If a Server Interruption Occurs

This table lists types of server interruptions and their results:

Table 63: Analytic Services Recovery Handling

Type of Interruption	Result
• Power loss on server machine • Operating system crash • Server stopped with Ctrl + C keys	Server stops. When you restart the server, Analytic Services recovers the database.
• Operation cannot complete due to system limitations • Memory shortage • Out of disk space • Application stopped with Ctrl + C keys	Analytic Services performs fatal error handling. You may need to allocate more memory or disk space and restart the server.
Server crash	Analytic Services Exception Manager creates an error log of type `.xcp`. Server stops. When you restart the server, Analytic Services recovers the database.

Table 64 shows what you need to do if a server interruption occurs during a transaction. How Analytic Services recovers from an interruption depends on the transaction isolation level setting (committed or uncommitted access). For an explanation of the recovery processes see "Rollback with Committed Access" on page 1060 and "Rollback with Uncommitted Access" on page 1062.

Table 64: Recovery Procedures for Server Requests

Type of Request	Recommended Action
Lock (for spreadsheet update)	Issue the lock command again.
Send (spreadsheet update)	If Analytic Services issues an error, repeat the last send operation. If the spreadsheet has been lost or does not exist, and if you are using SSAUDIT spreadsheet logging, reload the *dbname*.atx file. For an explanation of how and why to use spreadsheet update logging, see "How to Use Spreadsheet Update Logging" on page 1073.
Calculation	Check the server and application logs to see where the calculation left off. For a review of methods, see "Viewing the Analytic Server and Application Logs" on page 997. Decide whether to start the calculation over. Repeat the last calculation.
Data load	Complete one of the following actions: • Repeat the last data load. For instructions, see Chapter 19, "Performing and Debugging Data Loads or Dimension Builds." • Load the error log. For instructions, see "Loading Dimension Build and Data Load Error Logs" on page 1018.

Table 64: Recovery Procedures for Server Requests (Continued)

Type of Request	Recommended Action
Arithmetic data load (adding to or subtracting from values in the database)	If the database is set to committed access, reload the data. (The transaction has been rolled back.) If the database is set to uncommitted access, the process is not as simple. Some of the data loaded. Therefore, if you reload all of the data, you receive incorrect results for the data values that loaded twice. Therefore, you must perform the following actions: • Clear the database. • Restore the database to its state before the load. • Rerun the load.
Restructure	The restructure is not complete. First, delete the temporary restructure files: `.pan`, `.inn`, and `.otn`. Repeat the last operation that caused a restructure.

Note: If the UPDATECALC parameter is set to FALSE, Analytic Services recalculates the entire database if an interruption occurs during a calculation. (The default value of the parameter is TRUE.)

How to Use Spreadsheet Update Logging

For extra protection against data loss and for spreadsheet audit information, Analytic Services provides a spreadsheet update logging facility. Enable this facility by using the SSAUDIT or SSAUDITR parameter in the `essbase.cfg` file on the server. You can specify SSAUDIT for all databases on the server or for individual databases. For information on the `essbase.cfg` file and for syntax information on SSAUDIT and SSAUDITR, see the *Technical Reference*.

Analytic Services handles recovery under normal situations. However, sometimes you may want to load the spreadsheet update log manually. For example, if you have restored from a recent backup and do not want to lose changes made since the backup was made or you experience a media failure, you can recover transactions from the update log. To do so, use the Analytic Services command-line facility, ESSCMD, from the server console.

The following ESSCMD command sequence loads the update log:

```
LOGIN hostnode username password
SELECT application_name database_name
LOADDATA 3 filepath:application_name.ATX
EXIT
```

To simplify the process of loading the update log, prepare a batch file as described in "Using Script and Batch Files for Batch Processing" on page 1403.

When SSAUDIT or SSAUDITR is specified, Analytic Services logs spreadsheet update transactions chronologically. Analytic Services uses two files:

- *dbname*`.atx` stores spreadsheet update transactions as a unit that can be used as the input source for data loads.
- *dbname*`.alg` contains historical information for each transaction, such as user name, date, and timestamp, and the number of transaction rows from the `.atx` file.

Both files are stored on the server.

The spreadsheet update log can get quite large; even if you are using SSAUDITR, Analytic Services clears the log only after you back up data. If spreadsheet update activities are frequent in an application, you may want to manually delete the log periodically.

When a database is started after a shutdown, if spreadsheet logging is enabled, Analytic Services writes the following message to the database log:

```
Starting Spreadsheet Log
volumename\application_directory\application_name\
database_name\database_name.atx For Database database_name
```

An example of the message follows:

```
Starting Spreadsheet Log \ESSBASE\app\app1\sample\sample.atx for
database sample
```

To ensure successful spreadsheet update logging, stop and restart the application after either of the following:

- Any operation that causes a restructure. See Chapter 52, "Optimizing Database Restructuring."
- Running any of the following ESSCMD commands:

  CREATEAPP
  CREATEDB
  COPYDB
  RENAMEDB

Analytic Services ensures that if you enable spreadsheet logging, updates cannot take place without being logged. If Analytic Services cannot write to the update log for any reason, Analytic Services stops the transaction and issues an error message.

SSAUDIT and SSAUDITR are available only from the `essbase.cfg` file.

Considering Hybrid Analysis Issues

Hybrid Analysis offers a means of integrating a relational database with a multidimensional database so that lower level members and their associated data remain in the relational database while upper level members and their associated data reside in the Essbase database. This option presents additional issues regarding data consistency and integrity.

For information on ensuring that the data is correct in all locations, see "Managing Data Consistency" on page 302.

Chapter 47

Backing Up and Restoring Data

This chapter describes how to back up a database and lists the Analytic Services files that are essential to restoring a database from backups.

This chapter includes the following sections:

If you are migrating from a previous release of Analytic Services, see the *Essbase Analytic Services Installation Guide*.

Database Backups

A key part of a database maintenance routine includes regular backups of Analytic Services data. It is important to integrate regular database backups into production server maintenance.

The frequency of backups is dependent upon the volatility of the database and server environment, as well as upon the demand for quick database restores in the event of server crashes.

There are two methods of backing up a database:

- Preparing the database for file system backup
- Exporting, which makes a copy of data in a text format

This section tells you which files should be backed up regularly and describes each backup method.

Files to Back Up

You should regularly back up the server, application, and database files listed in Table 65:

Table 65: Files to Back Up

File	Location
ess*n*.ind	\essbase\app*appname**dbname*
ess*n*.pag	\essbase\app*appname**dbname*
dbname.esm	\essbase\app*appname**dbname*
dbname.tct	\essbase\app*appname**dbname*
dbname.ind	\essbase\app*appname**dbname*
dbname.app	\essbase\app
dbname.db	\essbase\app*appname**dbname*
x.lro	\essbase\app*appname**dbname*
dbname.otl	\essbase\app*appname**dbname*
essbase.sec	\essbase\bin
essbase.bak	\essbase\bin
essbase.cfg	\essbase\bin
Database object files such as: .otl .csc .rul .rep .eqd and .sel	\essbase\app*appname**dbname*
ESSCMD or MaxL scripts	No defined storage location

It is important to back up all .ind and .pag files related to a database because a single database can have multiple .ind and .pag files. Remember, the Agent should be shut down before the essbase.sec file is backed up.

For a full list and description of all Analytic Services files, see "Understanding How Analytic Services Files Are Stored" on page 950.

File System Backup

A common method of creating database backups is by doing a file system backup of the Analytic Server. You can perform the backup using the file system backup software of your choice. You can back up specific directories or files, or you can back up the entire Analytic Services directory structure. Be sure to back up data on every disk volume Analytic Services uses. For information about data storage on multiple volumes, see "Specifying Disk Volumes" on page 1038.

In most cases, backups occur after Analytic Services applications and databases, as well as the Agent, are shut down. If any Analytic Services databases must be up and running at the time of backup, follow these steps:

1. Place the database in read-only mode. See "Placing a Database in Read-Only Mode" on page 1079.
2. Perform the backup. See "Performing a Backup" on page 1081.
3. Return the database to read-write mode. See "Returning a Database to Read-Write Mode" on page 1081.

Placing a Database in Read-Only Mode

You can prepare a database for backup when the database must remain running during the backup process. Placing the database in read-only (or "archive") mode protects the database from updates during the backup process. After you perform the backup using the third-party backup utility of your choice, you then return the database to read-write mode.

➤ To place a database in read-only mode, use either of the following methods:

Tool	Instructions	For more information
MaxL	**alter database begin archive**	*Technical Reference*
ESSCMD	BEGINARCHIVE	*Technical Reference*

Note: If you try to cancel the BEGINARCHIVE ESSCMD command or the **alter database begin archive** MaxL statement and you receive a "can't cancel" message, the system is most likely in the final stage of writing items to the drive and has reached the point where the operation cannot be cancelled.

The begin-archive utility performs the following tasks:

- Commits any modified data to disk.
- Switches the database to read-only mode.
- Reopens the database files in shared, read-only mode.
- Creates a file containing a list of files that need to be backed up. By default, the file is called `archive.lst`. This file is stored in the *ARBORPATH*`\app\appname\dbname` directory.

If a user tries to modify data during the backup process, an error message informs the user that data is in read-only mode for backup.

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

CAUTION: If you back up data without using a begin-archive utility, make sure that all Analytic Services applications are closed and that all users are logged off during the backup process. Otherwise, you risk corrupting the database.

Performing a Backup

After putting the database in read-only mode, you are ready to perform the backup.

➤ To back up data, use a third-party backup utility to back up the files listed in `archive.lst` and the files listed in "Files to Back Up" on page 1078. Alternatively, you can back up the entire Analytic Services directory structure.

For information on restoring files from backup, see "Restoration of Data from Backups" on page 1085.

Returning a Database to Read-Write Mode

After performing the backup, return the database to read-write mode.

➤ To return the database to read-write mode, use either of the following methods:

Tool	Topic	Location
MaxL	**alter database end archive**	*Technical Reference*
ESSCMD	ENDARCHIVE	*Technical Reference*

The end-archive utility performs the following tasks:

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

Note: You must use the end-archive utility to put the database back into read-write mode, even if you shut down and restart the database. The end-archive utility does not restart the database.

Export Backups

You can back up data by exporting it. Exporting data copies it to a text file that you specify; it does not compress data. The export file contains data only and does not include control, outline, or security information.

You might consider exporting data for the following reasons:

- To transfer data across platforms
- To back up only a certain portion of the data; for example, level 0 blocks
- To create an exported file in text format, rather than binary format

Note: You can export subsets of data by creating reports. For a discussion and examples of the process, see "Exporting Data Using Report Scripts" on page 739.

Understanding the Advantages and Disadvantages of Exporting Data

Using export to back up data provides the following advantages:

- You can use the resulting text files to load data from the source database into databases on other platforms.

  When loading an export file into a database, it is important that the database outline contains all the members found within the export file. If not, the load will fail. Also, if the outline changes between the time that the export file is created and reloaded (and the new outline contains all the members found within the export file), the load time might be significantly higher than if the outlines were identical.

- During an export, data integrity is verified because every block is checked to confirm whether corresponding page and index files match.
- You can reduce fragmentation in a database by exporting data into a text file, clearing all data from the database, and reloading the text file.
- You can export a database in column format from Administration Services or MaxL. Then, you can use a rules file to load the column-formatted file. Using column format is helpful when you need to manipulate the export file.

Using export to back up data provides the following disadvantages:

- Because dynamic calculations are not executed at the time of the export, only stored data and data from previously calculated Dynamic Calc And Store members are included in the export.
- At the time of a database export, Analytic Services users cannot write to the database. Users receive an error message if they try to write to the database during an export. After an export has started, users can do read operations. Exports of large databases require considerable amounts of time, time during which users can only read the data.

Exporting Data

➤ To export data, use either of the following methods:

Tool	Instructions	For More Information
Administration Services	Exporting Databases	*Essbase Administration Services Online Help*
Report Writer	Use a Report Writer script to export selected data.	"Exporting Data Using Report Scripts" on page 739
ESSCMD	EXPORT or PAREXPORT	*Technical Reference*
MaxL	**export data**	*Technical Reference*

Note: To improve export performance, you can export data in parallel to a specified number of files.

All methods require the same basic information:

- The amount of data to export:
  - All data
  - Level 0 blocks only (blocks containing only level 0 sparse member combinations. Note that these blocks may contain data for upper level dense dimension members.)
  - Data from input blocks only (blocks containing data from a previous data load or spreadsheet Lock & Send)

- Whether to export data in a columnar or non-columnar format

  To facilitate loading the exported data into a relational database, export the data in columns. In each row, the columnar format displays a member name from every dimension. Names can be repeated from row to row.

  The 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.

- The export data file names

Exporting Files Larger Than 2 GB

Some file management systems do not support text files larger than 2 GB. On any operating system, if Analytic Services anticipates that an export file exceeds 2 GB, it creates two or more export files, as needed.

When Analytic Services creates multiple export files, it uses the requested file name for the main file. An underscore and a sequential cardinal number are appended to the names of the additional files, starting with _1. For example, if the requested file name is `expJan.txt` and the exported data would exceed 4 GB, Analytic Services creates three files, naming them `expJan.txt`, `expJan_1.txt`, and `expJan_2.txt`. Exported data files can be reloaded in any sequence.

Reloading Exported Data

➤ To reload exported data, use any of the following methods:

Tool	Topic	Location
Administration Services	Performing a Data Load or Dimension Build	*Essbase Administration Services Online Help*
ESSCMD	IMPORT	*Technical Reference*
MaxL	**import data**	*Technical Reference*

When you reload data that has been exported, Analytic Services marks the data as input data. If you reload data exported from level 0 blocks or input blocks, you must recalculate the database after reloading. When you recalculate the database, Analytic Services recalculates every data block.

If you export *all* data in a database and then reload, Analytic Services marks all blocks in the database as input blocks. Consequently, if you try to clear data, no data is cleared because the database contains no non-input blocks.

When you reload data that has been exported, Analytic Services also marks the data blocks as dirty. If you had calculated the database prior to exporting it, to save time during the next calculation, you should set the status of the blocks as clean. If you had not calculated the database prior to exporting it, you do not need to set the status of the blocks as clean.

➤ To clean the status of the blocks in a database after exporting all data and reloading, you can run the following calculation script:

```
Set ClearUpdateStatus Only;
Calc All;
```

Restoration of Data from Backups

To restore a database, replace the files on disk with the corresponding files from backup. See "Files to Back Up" on page 1078 for a list of files that should be backed up on a regular basis.

The application should be stopped, unless you are restoring from an export file. In that case, ensure the application is not accepting client connections.

Changing Frequency of Backup File Comparisons

Essbase now compares the security backup file essbase.bak to the security file `essbase.sec` at specified intervals instead of only when Analytic Server starts.

➤ To change the frequency of these backup file comparisons, use either of the following methods:

Tool	Instructions	For more information
Administration Services	Enter the time interval in the **Check for inactivity every** option of the Security tab when you edit Analytic Server properties.	*Essbase Administration Services Online Help*
MaxL	**alter system sync security_backup**	*Technical Reference*

Note: You can manually update the security backup file at any time using Administration Services. See "Updating the Security Backup File" in *Essbase Administration Services Online Help*.

Review these facts before changing the interval value:

- In Administration Services, the same check box manages how often the security backup file is checked against the security file and how often user inactivity is checked.
- By default, the value is five minutes. Five minutes is the recommended setting to ensure that the security backup file is checked frequently enough to capture security changes. Five minutes is also the recommended value for the inactivity check.
- If you set the value to zero, the inactivity check is disabled and the `essbase.bak` is compared to `essbase.sec` every five minutes (and, as always, updated if necessary).
- Enter a larger value if your security file does not need to be updated frequently. Enter a smaller value if performance is not an issue.

Essbase always updates the backup file if it does not match the security file when the two files are compared, regardless of which tool is used to trigger the comparison. The backup file is updated only if a difference exists between the security file essbase.sec and the security backup file `essbase.bak`.

CAUTION: If Essbase stops running unexpectedly for any reason, such as a freeze or crash, or as the result of terminating a process, do not restart Analytic Server until you copy the backup file `essbase.bak` to the security file `essbase.sec`. If you do not perform the copy first, when Analytic Server starts, Essbase notes that essbase.sec is corrupt, creates an empty security file and copies it to `essbase.bak`, thus destroying the backup of your security information.

Essential Database Files

These files are all key components of an Analytic Services database:

Table 66: Essential Database Files

File	Description
`ess`*`n`*`.pag`	Analytic Services data file
`ess`*`n`*`.ind`	Analytic Services index file
`dbname``.esm`	Analytic Services Kernel file that contains control information used for database recovery
`dbname``.tct`	Transaction control table
`dbname``.ind`	Free fragment file for data and index free fragments
`dbname``.otl`	Outline file, which does not store data but does store all metadata for a database and defines how data is stored

If there is a problem with any one of these files, the entire database becomes corrupted. The database must then be restored from backups or reloaded from exports (see “Database Backups” on page 1077.)

There have been cases in which database files have become corrupted. In such situations, the database is not able to start up on Analytic Server. Therefore, no data can be reloaded to restore the database. In these cases, the only way to restore the database is to delete all the following files:

- ess*n*.pag
- ess*n*.ind
- *dbname*.esm
- *dbname*.tct
- *dbname*.ind

After the files are deleted, restart the database and reload from data files or from export files created prior to the corruption.

Chapter 48

Using MaxL Data Definition Language

This chapter describes MaxL DDL, the data definition language for Analytic Services. MaxL is a flexible way to automate Analytic Services administration and maintenance tasks. MaxL DDL improves upon ESSCMD in that it is a multi-dimensional access language for Analytic Services. Using MaxL, you execute Analytic Services operations using a language-like interface, rather than a series of commands with complicated arguments.

MaxL scripts can be developed flexibly to accommodate many uses, and the language is easy to learn. MaxL includes a Perl module that enables you to embed its statements in Perl programs.

This chapter contains the following sections:

- "The MaxL DDL Language" on page 1089 discusses core language elements.
- "The MaxL Shell" on page 1096 describes how to use the shell interface.
- "The MaxL Perl Module" on page 1103 describes the optional Perl interface.

For comprehensive coverage of MaxL syntax, see the MaxL DDL section of the *Technical Reference*.

The MaxL DDL Language

The MaxL data definition language is an interface for making administrative requests to Analytic Services using *statements*.

Using MaxL, you can easily automate administrative operations on Analytic Services databases. You can write MaxL scripts with variables to make them customizable and re-usable.

In order for Analytic Services to receive and parse MaxL statements, you must "pass" them to the Analytic Server using either the MaxL Shell (`essmsh`), Administration Services, or a customized Perl program that uses the MaxL Perl Module.

Use this section to learn more about the language:

- "Overview of Statements" on page 1090
- "Components of Statements" on page 1090
- "Analysis of Statements" on page 1095

Overview of Statements

All MaxL DDL scripts and interactive sessions consist of a login and a sequence of statements. Each statement is terminated by a semicolon.

Statements consist of grammatical sequences of keywords and variables. A statement looks similar to an English sentence; for example,

```
create or replace user <user-name> identified by <password>;
```

MaxL statements always begin with a verb, such as *create* or *alter.* Verbs indicate what type of operation you want to perform. After the verb, you specify an object. Objects, such as *database* or *user*, indicate the Analytic Services elements you are working with. After the object specification, the rest of the statement is for giving more details about the action you wish to perform. This is done using a grammatically correct sequence of statement-parts, or *tokens*.

To get an overall picture of the grammar requirements and the verb-object structure of MaxL statements, see the MaxL DDL section of the *Technical Reference*.

Components of Statements

The MaxL parser recognizes and requires an ordered presentation of *tokens,* and these are the components of statements. A token is a space-delimited sequence of valid characters that is recognized by MaxL as a single readable unit. Tokens can be any of the following:

- Keywords
- Names

- Strings
- Numbers

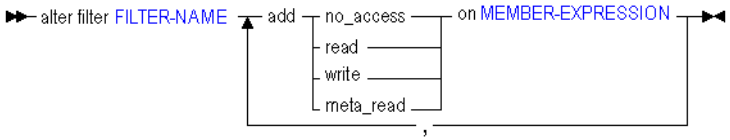
Keywords

Keywords are the reserved words that make up the MaxL vocabulary. These include verbs, objects, and other words needed to construct statements. Keywords in MaxL are independent of your data: conversely, all other MaxL tokens (names, for example) must be defined by you.

The MaxL parser expects to see MaxL keywords and other tokens in their correct grammatical order, as diagrammed in the MaxL DDL section of the *Technical Reference*.

In the following sample syntax diagram from the *Technical Reference*, only the lower-cased words in the diagram represent keywords. The other elements are placeholders for names or values that you provide.

Figure 234: Example of MaxL Syntax Diagram: Alter Filter



Note: Keywords are not case-sensitive; the use of lower case for keywords is a documentation convention. For more information about how to interpret the diagrams, see “How to Read MaxL Railroad Diagrams” in the online *Technical Reference*.

Names

Names in MaxL are used to uniquely identify databases and database objects, such as users, applications, or filters.

Rules for Names

Unless you enclose a MaxL name within single quotation marks, a MaxL name is a string that must begin with an alphabetic character or the underscore. Names that are not enclosed in quotation marks may contain only alphabetic characters, numbers, and the underscore.

When enclosed in single quotation marks, a name may contain white space and any of the following special characters:

. , ; : % $ " ' * + - = < > [ ] {} ( ) ? ! / \ | ~ ` # & @ ^

Note: Any name that is also a MaxL keyword must be enclosed in single quotation marks. For a list of keywords, see the "Reserved Words List in the MaxL DDL" section of the *Technical Reference*.

Examples:

The following application names *do not* require single quotation marks:

Orange
Orange22
_Orange

The following application names *do* require single quotation marks:

`Orange County` (contains a space)
`22Orange` (begins with a number)
`variable` (is a MaxL keyword)

Types of Names

Some Analytic Services objects have single names, and some require compound names known as *doubles* and *triples*, which express the nesting of namespaces. The three possible ways to name Analytic Services objects are with a singleton name, a double, or a triple.

A *singleton name* is a name that can be meaningful in a system-wide context: the object to which it refers may be global to Analytic Services, or, for whatever reason, it needs no application or database context to be specified. For example, an application has a singleton name because it does not need to be considered in the context of another application or a database.

A *double* is two names connected by a period, and a *triple* is three names connected by two periods. Doubles and triples show the inherited namespace of the named entity. For example, a database is usually identified using two names. The first name identifies the application in which the database resides. The second name is the database's own name. Therefore, a database name could look like this:

```
Sample.Basic
```

Database objects, such as filters, are usually identified using triple names: the first two names identify the application and database, and the third name is the object's own name. Therefore, a filter name could look like this:

```
sample.basic.filter3.
```

The following table shows what type of name is required for the most common objects, and provides an example of the name used in a statement.

Object	Name	Example
User	singleton	`create user **Fiona** identified by 'password';`
Group	singleton	`alter user Fiona add to group **Managers**;`
Host	singleton	`drop replicated partition Samppart.Company from Sampeast.East at **EastHost**;`
Application	singleton	`create application '**&New App**';`
Database	double	`display database '**&New App**'.**testdb**;`
Calculation	triple	`drop calculation **Sample.basic.'alloc.csc'**;`
Filter	triple	`display filter row **sample.basic.filter1**;`
Function (local)	double	`drop function **sample.'@COVARIANCE'**;`
Function (global)	singleton	`create function '**@JSUM**' as 'CalcFnc.sum';`
Location alias	triple	`drop location alias **Main.Sales.EasternDB**;`
Role	singleton	`grant **designer** on database Sample.basic to Fiona;`
Substitution variable	singleton	`alter system add variable **Current_month**;` `alter system set variable **Current_month** July;`

Disk volume	singleton to define, triple to display	`alter database AP.main1 add disk volume **G**;`
		`alter database AP.main1 set disk volume **G** partition_size 200mb;`
		`display disk volume **sample.basic.C**;`

Strings

Strings are used in MaxL statements to represent the text of comments, member expressions, calculation scripts, and file references. Strings can begin with any valid character. As with names, strings containing whitespace or special characters must be enclosed in single quotation marks.

See "Rules for Names" on page 1091 for a list of valid special characters.

The following table shows examples of statement elements that are strings:

Type of String	Example
Password	`create user Fiona identified by sunflower;`
Comment	`alter group Financial comment 'Reports due July 31';`
Member expression	`create filter sample.basic.filt1 read on 'Sales, @ATTRIBUTE(Bottle)';` `create or replace replicated partition sampeast.east area '@IDESC(East), @IDESC(Qtr1)' to samppart.company mapped globally '("Eastern Region")' to '(East)';`

Body of a calculation	`execute calculation '"Variance"=@VAR(Actual, Budget);`
	`"Variance %"=@VARPER(Actual, Budget);'`
	`on Sample.basic;`

File reference	`spool on to '/homes/fiona/out.txt';`

Numbers

You use numbers in MaxL statements to change certain database settings in Analytic Services. For example, you can change cache and buffer sizes, or set system-wide intervals such as the number of days elapsing before users are required to change their passwords. To change numeric settings, you can use positive integers, positive real numbers, and zero. Decimals and scientific notation are permitted.

Examples:

1000
2.2
645e-2

For size settings, units must follow the number. Spaces in between numbers and units are optional. Units are case-insensitive, and may include the following: B/b (bytes), KB/kb (kilobytes), MB/mb (megabytes), GB/gb (gigabytes), and TB/tb (terabytes). If no units are given, bytes are assumed.

Examples:

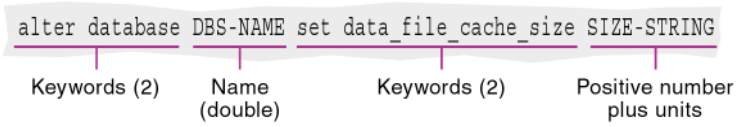
1000 b
5.5GB
645e-2 mb
145 KB
2,000e-3TB

Analysis of Statements

This section helps you review what you have learned about statements by illustrating some MaxL statements and their components, in template form. In the diagrams, lower-cased words are keywords, and words in upper-case are intended to be replaced with the appropriate values, as shown in the example following each illustration.

Altering a Database

Figure 235: MaxL statement to change data file cache size

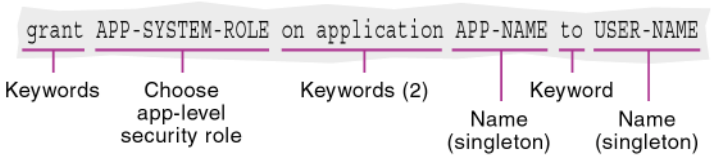


Example:

```
alter database Sample.Basic set data_file_cache_size 32768KB;
```

Granting a Permission

Figure 236: MaxL statement to grant application permissions to a user

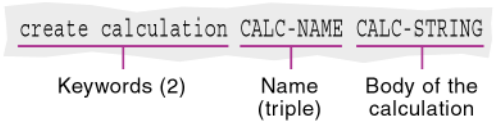


Example:

```
grant designer on application Sample to Fiona;
```

Creating a Calculation

Figure 237: MaxL statement to create a stored calculation



Example:

```
create calculation sample.basic.varcalc
'"Variance"=@VAR(Actual, Budget);
"Variance %"=@VARPER(Actual, Budget);'
;
```

The MaxL Shell

This section shows you how to get started using the most of the features of the MaxL Shell. For more complete information and examples, see the MaxL DDL > MaxL Shell section of the *Technical Reference*. This section contains the following topics:

- “Starting the MaxL Shell” on page 1097
- “Logging In to Analytic Services” on page 1100

This section does not discuss the Administration Services MaxL Script Editor. If you are using MaxL Script Editor, you can skip the rest of this chapter and refer instead to the *Essbase Administration Services Online Help*.

Starting the MaxL Shell

The MaxL Shell can be invoked to take input in any of the following ways:

- Interactively, from the keyboard
- From a MaxL script file (statements are read from the file specified on the command line)
- From standard input that is piped to the MaxL Shell from the output of another program

The MaxL Shell also accepts any number of command-line arguments at invocation time. These can be used with positional parameters to represent any name, or a password.

This topic contains the following sections:

Starting the Shell for Interactive Input

➤ To enter MaxL statements interactively at the command line, simply invoke the shell at your operating-system prompt. Text you type is indicated by bold text.

For example,

```
essmsh

Analytic Services MaxL Shell Release 7.0.0
(c) Copyright 2000-2003 Hyperion Solutions Corporation.
All rights reserved.
```

- To enter MaxL statements interactively after logging in at invocation time, use the -l flag. For example,

```
essmsh -l Admin password

Analytic Services MaxL Shell Release 7.0.0
(c) Copyright 2000-2003 Hyperion Solutions Corporation.
All rights reserved.

49 - User logged in: [Admin].
```

- To enter MaxL statements interactively and also supply command-line arguments to represent variables you will use in your interactive session, use the -a flag. For example,

```
essmsh -a Admin password Sample Basic

Analytic Services MaxL Shell Version 7.0.0
(c) Copyright 2000-2003 Hyperion Solutions Corporation.
All rights reserved.

login $1 $2;

49 - User logged in: [admin].

alter database $3.$4 enable aggregate_missing;

72 - Database altered: ['sample'.'basic'].
```

  In the above example, $1, $2, $3, and $4 are positional parameter variables: they represent whatever arguments were entered after **essmsh** at invocation time, in the order they were entered.

Starting the Shell for File Input

- To invoke the MaxL Shell to take input from a MaxL script file, type **essmsh** followed by the name of a MaxL script in the current directory, or, the full path and file name of a MaxL script in another directory.

  If you provide only a file name, the MaxL Shell assumes that the file is in the current directory (the directory the operating-system command prompt was in when essmsh was invoked). In the following invocation example, the file `maxlscript.msh` must be in C:\.

```
C:\> essmsh maxlscript.msh
```

If the MaxL script is not in the current directory, provide a path to the MaxL script. You can use absolute paths or relative paths.

For example,

```
$ essmsh ../hyperion/essbase/test.msh
```

Note: MaxL scripts are not required to have any particular file extension, or any file extension at all. This document uses `.msh`.

In UNIX shells, you should place single quotation marks around the path to avoid file-reading errors.

In a Windows command prompt, if the path to the script contains a space, you may have to use double quotation marks around the entire path and file name to avoid file-reading errors.

Starting the Shell for Programmatic Input

➤ To invoke the MaxL Shell to take input from the standard output of another program or process, use the -i flag. For example,

```
program.sh | essmsh -i
```

The shell script `program.sh` may generate MaxL statements as output. The shell script's output is piped to `essmsh -i`, which uses that output as its input. This allows for efficient co-execution of scripts.

Windows Example Using -i Invocation

The following Windows batch script generates a login statement and a MaxL display statement as its output. The -i flag enables that output to be used by `essmsh`, the MaxL Shell, as input.

```
echo login admin password on localhost; display privilege
user;|essmsh -i
```

User Admin is logged in, all user privileges are displayed, and the MaxL session is terminated.

UNIX Example Using -i Invocation

The following portion of a shell script ensures that there are no applications on the system, by testing whether the **display application** statement returns zero applications.

```
if [ $(echo "display application;" |
essmsh -l admin password -i 2>&1 |
awk '/Records returned/ {print $7}' ) != "[0]." ]
then
   print "This test requires that there be no applications."
   print "Quitting."
   exit 2
fi
```

Here is how the above example works:

1. MaxL grammar is piped to a MaxL Shell invocation and login, as the output of the UNIX `echo` command.
2. The results of the MaxL session are tested by `awk` for pattern-matching with the MaxL status message you would get if you entered **display application** on an empty system: `Records returned: [0]`.
3. `Awk` matches the string `'Records returned: '`, and then it checks to see if that is equal to `'[0].'`
4. If `$7` (a variable representing the fifth token `awk` finds in the status string) is equal to `'[0].'`, then there are no applications on the system; otherwise, `$7` would equal `'[1].'` or whatever number of applications exist on the system.

For more information and examples on invocation options, see the MaxL DDL > MaxL Shell section of the *Technical Reference*. Invocation information is also contained in the essmsh "man page." To view the man page, enter `essmsh -h | more` at the operating-system command prompt.

Logging In to Analytic Services

The MaxL language interpreter requires a connection to an Analytic Services session before it can begin parsing MaxL statements. Use the MaxL Shell to establish the connection to Analytic Services.

- To log in to Analytic Services after the command shell has been started, use the shell's **login** grammar. Text you type is indicated by bold text.

  For example,

  ```
  essmsh

  Analytic Services MaxL Shell Release 7.0.0
  (c) Copyright 2000-2003 Hyperion Solutions Corporation.
  All rights reserved.

  MAXL>login Fiona identified by sunflower on hostname;
  ```

  If a host name is not specified, localhost is assumed.

- To log in when you invoke the shell, use the -l option. To log in to a server besides `localhost` at invocation time, use the -s option. To set a message level, use -m. For example,

  ```
  essmsh -l fiona sunflower -s myHost -m error
  ```

  Note: You can log out and change users without quitting the shell.

  For more information about MaxL Shell invocation options, see the MaxL DDL > MaxL Shell section of the *Technical Reference*.

Command Shell Features

The MaxL Shell includes command-line argument processing, environment variable processing, nesting of MaxL scripts, and shell escapes. These features offer the flexibility needed to create a highly automated Analytic Services production environment.

For complete information on the syntax and usage of all MaxL Shell features, see the MaxL DDL > MaxL Shell section of the *Technical Reference*.

Nesting MaxL Scripts

As a database administrator, you may wish to save your separate automated tasks in several MaxL scripts, rather than executing many operations from a single script. Putting the pieces together is a simple task if you know how to reference one MaxL script from another.

➤ To reference or include other MaxL scripts within the current MaxL session, use the following MaxL Shell syntax:

```
msh <scriptfile>;
```

Spooling Output to a File

You can create a log file of all or part of a MaxL session and its associated messages by spooling output to a file.

➤ To record a MaxL session, complete the following tasks:

1. Log in to Analytic Services. For example,

   ```
   login fiona sunflower;
   ```

2. Begin spooling output, using **spool on to <filename>**.

   For example, `spool on to 'c:\\output\\display.txt';`

3. Enter as many MaxL statements as you want recorded.
4. Stop spooling output, using **spool off**;

MaxL statements and their output are logged to the output file when you issue the spool command, either in an interactive session or in a script. However, MaxL *Shell* commands and output are logged only if you spool during an interactive session. The MaxL Shell commands and output are ignored in log files created from script sessions. Additionally, output from any operating-system commands you may have included is ignored in the log files of both interactive and script sessions.

Including Operating-System Commands

You can issue operating-system commands directly from a MaxL session. The operating-system output becomes part of the MaxL Shell output. When the operating system finishes executing commands, it returns control to `essmsh`.

➤ To escape to the operating system from within a MaxL session, use **shell**. For example, try running the UNIX **date** command from a MaxL script.

```
shell date;
```

- To escape to ESSCMD from within a MaxL session:

```
shell esscmd '../scripts/test.scr';
```

Using Variables to Customize MaxL Scripts

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, or passwords. You can use variables in MaxL scripts and during interactive sessions. Using variables in MaxL scripts eliminates the need to create customized scripts for each user, database, or host. Variables can be environment variables (for example, `$ARBORPATH`, which references the Analytic Services installation directory), positional parameters (for example, $1, $2, and so on), and locally defined shell variables. A variable always begins with a $ (dollar sign) when you reference it.

For more information about using variables in the MaxL Shell, see the MaxL DDL > MaxL Shell section of the *Technical Reference*.

Stopping the MaxL Shell

You can log out of a MaxL session, or log in as another user, without quitting the shell. You should include a logout statement at the end of MaxL scripts. It is not necessary to exit at the end of MaxL script files, or after a session using stream-oriented input from another program's output.

- To log out without exiting the MaxL Shell, enter

```
logout;
```

- To exit from the MaxL Shell after using interactive mode, enter

```
exit;
```

The MaxL Perl Module

With the aid of the MaxL Perl Module (Essbase.pm), you can embed the MaxL language within Perl programs, offering more programmatic control than is available in the shell.

Essbase.pm, located in the `Perlmod` directory, enables beginning or advanced Perl programmers to wrap MaxL statements in Perl scripts. In this way, database administration with MaxL becomes as efficient and flexible as your Perl programs are.

Using Perl with MaxL enables you to take advantage of these and other programmatic features while you administer Analytic Services databases:

- Conditional testing
- Inter-process communication
- Message handling
- E-mail notification
- Web scripting

The Perl Module (`Essbase.pm`), contains methods that enable you to pass MaxL statements by means of Perl. These methods are:

- **connect ()**, which establishes a connection to Analytic Services
- **do ()**, which tells Perl to execute the enclosed MaxL statement
- **pop_msg ()**, which navigates through the stack of returned MaxL messages
- **fetch_col ()**, **fetch_desc ()**, and **fetch_row ()**, which retrieve information from MaxL display output tables
- **disconnect ()**, which terminates the connection to Analytic Services

To make the Perl methods available to Analytic Services, you must include a reference to `Essbase.pm` in your Perl program. Place the following line at the top of each Perl script:

```
use Essbase;
```

Perl is not difficult to learn, especially if you have prior knowledge of UNIX shells or other programming languages. To download Perl and learn more about Perl, visit the Comprehensive Perl Archive Network Web site at `http://www.cpan.org/`.

For information and examples about installing and using the MaxL Perl Module, see the MaxL DDL section of the *Technical Reference*, and also the README file located in the PERLMOD directory of your Analytic Services installation.

Part IX

Optimizing Analytic Services

This part describes how to optimize Analytic Services:

- Chapter 49, "Monitoring Performance," describes how to use diagnostic tools to analyze and solve problems, check Analytic Services configuration, and analyze Analytic Services performance.
- Chapter 50, "Improving Analytic Services Performance," supplies a list of settings and suggested and default values. These settings all affect performance.
- Chapter 51, "Optimizing Analytic Services Caches," describes how to size the Analytic Services index cache, data file cache, and data cache.
- Chapter 52, "Optimizing Database Restructuring," explains the conditions under which Analytic Services must perform database restructuring, and how you can design your outline to minimize this.
- Chapter 53, "Optimizing Data Loads," describes how to debug problems with data loads and how to optimize your data loads so that Analytic Services can load the data more quickly.
- Chapter 54, "Optimizing Calculations," describes how to make your calculation scripts execute more quickly.
- Chapter 55, "Optimizing with Intelligent Calculation," describes how to use intelligent calculation to make your calculation scripts execute more quickly.
- Chapter 56, "Optimizing Reports and Other Types of Retrieval," describes ways to generate your reports more quickly, and how to optimize data extraction.

Note: For information about performance and partitions, see "Performance Considerations for Replicated Partitions" on page 246 and "Performance Considerations for Transparent Partitions" on page 253.

Chapter 49

Monitoring Performance

This chapter describes Analytic Server diagnostic information and tells you how to display that information. Use this information to monitor Analytic Services configuration or performance.

This chapter contains the following sections:

For a comprehensive discussion of activity, error and exception logs, see Chapter 43, "Monitoring Data, Applications, and Databases." For information about specific error messages, see the *Error Messages Guide* in `\docs\errmsgs\erhelp.htm` in the Analytic Services installation.

Finding Diagnostic Information

Analytic Services provides performance information dialog boxes at the Analytic Server, application, and database level. View performance information before completing any of these tasks:

- Preparing to migrate data
- Adding users
- Analyzing performance problems
- Performing other administrative tasks

Analytic Services displays information on a snapshot basis; to see the latest information, click the Refresh button for a new snapshot. The Refresh button, if it is displayed in a window or dialog box, refreshes every tab in the window or dialog box, not just the current tab.

For a comprehensive discussion of Analytic Server, application, and outline logs, see "Analytic Server and Application Logs" on page 979.

The following sections provide detailed procedures for accessing information about Analytic Servers, applications, and databases that are commonly used to diagnose performance or other issues.

Viewing Analytic Server Information

You can view information about the Analytic Server license, configuration, operating system, disk drives, and applications for Analytic Server.

➤ To view Analytic Server information, use any of the following methods:

Tool	Topic	Location
Administration Services	Monitoring Analytic Servers	*Essbase Administration Services Online Help*
MaxL	**display application**	*Technical Reference*
ESSCMD	GETAPPSTATE GETPERFSTATS	*Technical Reference*

Viewing Application Information

You can view application information to identify which databases are running in the application and to check access, security, and start-up information.

➤ To view application information, use any of the following methods:

Tool	Topic	Location
Administration Services	Monitoring Applications	*Essbase Administration Services Online Help*
MaxL	**display application**	*Technical Reference*
ESSCMD	GETAPPSTATE GETPERFSTATS	*Technical Reference*

Viewing Database Information

You can view information about database storage, database statistics, and lock contention. This information may help you identify activities or operations that affect performance.

➤ To view database information, use any of the following methods:

Tool	Instructions	For More Information
Administration Services	Monitoring Databases	*Essbase Administration Services Online Help*
MaxL	**display database**	*Technical Reference*
ESSCMD	• For general information: GETDBINFO • For database storage information: GETDBSTATE • For currency information: GETCRDBINFO • For database statistics: GETDBSTATS • For run-time information: GETDBINFO	*Technical Reference*

Monitoring Application/Database Status

You can view the start/stop status of applications and databases that you are authorized to use.

➤ To view application/database status, use any of the following methods:

Tool	Instructions	For More Information
Administration Services	Viewing Application/Database Status	*Essbase Administration Services Online Help*
ESSCMD	GETAPPINFO GETDBINFO	*Technical Reference*

Monitoring User Sessions and Requests

You can monitor active user sessions for an Analytic Server, application, or database. If you have Supervisor or Application Designer permissions, you can disconnect a user session or terminate a specific request made during a session. for more information, see "Disconnecting Users and Terminating Requests" on page 856.

➤ To monitor user sessions and requests, use any of the following methods:

Tool	Instructions	For More Information
Administration Services	Viewing Active User Sessions	*Essbase Administration Services Online Help*
MaxL	**display session alter system**	*Technical Reference*

Monitoring Applications from the Operating System

Each application that is loaded is an open task or process in the operating system. You can use the operating system to view application tasks or processes:

- On Windows platforms, the application is displayed in an application server window. You can view application activities as they occur in this window. When the Agent starts an application, a new icon is displayed in the task bar. You can double-click the icon to view the application server window.
- On UNIX platforms, the application server is often a background process. When the application starts, ESSBASE starts the ESSSVR process. In order to see activities, you can route all messages to a file with the `tail -f log` command, where *log* is the name of a file that you specify.
- You can also view a snapshot of the Analytic Server or application log using Administration Services. See the *Essbase Administration Services Online Help* for details about viewing, filtering, searching, and analyzing logs. See "Analytic Server and Application Logs" on page 979 for a comprehensive discussion of server and application logs.

Chapter 50

Improving Analytic Services Performance

You can improve Analytic Services performance with these basic techniques:

Recognizing Basic Design Issues That Affect Optimization

Use the following list to identify basic design issues that affect optimization outside this volume:

- For an introduction to basic concepts and how they relate to optimized performance, see Chapter 4, "Basic Architectural Elements."
- For a discussion of how to analyze a database design while it is still on paper and of how this analysis can aid optimization, see "Analyzing and Planning" on page 80.
- To understand how basic database outline issues affect performance, see "Designing an Outline to Optimize Performance" on page 100.

Resetting Databases to Increase Performance

You can periodically reset a database, and then reload it. Even if you reload a database very often, the main database files, `.pag` files, can grow unless you reset the database.

➤ To reset a database, use either of the following methods:

Tool	Topic	Location
MaxL	**alter database** *appname.dbname* **reset**	*Technical Reference*
ESSCMD	RESETDB	*Technical Reference*

Using Database Settings to Customize for Maximum Performance

You can customize Analytic Services for maximum performance, using database settings at the database level in Administration Services, ESSCMD, or MaxL.

Note: If you are migrating a database, see the *Essbase Analytic Services Installation Guide* for information about the default settings after migration.

The following sections list performance settings and describe how to adjust them.

- “Database Cache Settings” on page 1115
- “Database Disk Volumes Settings” on page 1116
- “Database Transaction Control Settings” on page 1117
- “Miscellaneous Database Settings” on page 1119

Database Cache Settings

The following table describes database cache settings and lists the location of the settings in Administration Services, MaxL, and ESSCMD.

Table 67: Database Cache Settings

Setting	More Information	Location in Administration Services, MaxL, ESSCMD
Index cache size	"Sizing the Index Cache" on page 1129	• Administration Services: Database Properties window, Caches tab • MaxL: **alter database** *appname.dbname* **set index_cache_size** *n* • ESSCMD: SETDBSTATEITEM 12
Data file cache size	"Sizing the Data File Cache" on page 1130	• Administration Services: Database Properties window, Caches tab • MaxL: **alter database** *appname.dbname* **set data_file_cache_size** *n* • ESSCMD: SETDBSTATEITEM 27
Data cache size	"Sizing the Data Cache" on page 1132	• Administration Services: Database Properties window, Caches tab • MaxL: **alter database** *appname.dbname* **set data_cache_size** *n* • ESSCMD: SETDBSTATEITEM 5
Index page size	Fixed size.	N/A
Cache memory locking	"Deciding Whether to Use Cache Memory Locking" on page 1127	• Administration Services: Database Properties window, Caches tab • MaxL: **alter database** *appname.dbname* **enable cache_pinning** • ESSCMD: SETDBSTATEITEM 26

For more information about these settings in Administration Services, see “Setting Database Properties” in *Essbase Administration Services Online Help*.

Database Disk Volumes Settings

The following table describes database disk volume settings and lists the location of the settings in Administration Services, MaxL, and ESSCMD.

Table 68: Database Disk Volume Settings

Setting	More Information	Location in Administration Services, MaxL, ESSCMD
Volume name	“Specifying Disk Volumes” on page 1038	• Administration Services: Database Properties window, Storage tab • MaxL: **alter database** *appname.dbname* **set disk volume** • ESSCMD: SETDBSTATEITEM 23 SETDBSTATEITEM 24
Partition size	“Specifying Disk Volumes” on page 1038	• Administration Services: Database Properties window, Storage tab • MaxL: **alter database** *appname.dbname* **set disk volume** • ESSCMD: SETDBSTATEITEM 23 SETDBSTATEITEM 24
File type	“Specifying Disk Volumes” on page 1038	• Administration Services: Database Properties window, Storage tab • MaxL: **alter database** *appname.dbname* **set disk volume** • ESSCMD: SETDBSTATEITEM 23

Table 68: Database Disk Volume Settings (Continued)

Setting	More Information	Location in Administration Services, MaxL, ESSCMD
Maximum file size	"Specifying Disk Volumes" on page 1038	• Administration Services: Database Properties window, Storage tab • MaxL: **alter database** *appname.dbname* **set disk volume** • ESSCMD: SETDBSTATEITEM 23

For more information about these settings in Administration Services, see "Setting Disk Volumes" in *Essbase Administration Services Online Help*.

Database Transaction Control Settings

The following table describes database transaction control settings and lists the location of the settings in Administration Services, MaxL, and ESSCMD.

For more information about database transaction control settings in Administration Services, see "Setting Data Integrity Options" in *Essbase Administration Services Online Help*.

Table 69: Database Transaction Control Settings

Setting	More Information	Location in Administration Services, MaxL, ESSCMD
Isolation level	"Understanding Isolation Levels" on page 1054	• Administration Services: Database Properties window, Transactions tab • MaxL: **alter database** *appname.dbname* **enable committed_mode** • ESSCMD: SETDBSTATEITEM 18

Table 69: Database Transaction Control Settings (Continued)

Setting	More Information	Location in Administration Services, MaxL, ESSCMD
Commit Blocks	"Understanding Isolation Levels" on page 1054	• Administration Services: Database Properties window, Transactions tab • MaxL: **alter database** *appname.dbname* **enable committed_mode** and **alter database** *appname.dbname* **set implicit_commit after** *n* **blocks** • ESSCMD: SETDBSTATEITEM 21
Commit Rows	"Understanding Isolation Levels" on page 1054	• Administration Services: Database Properties window, Transactions tab • MaxL: **alter database** *appname.dbname* **enable committed_mode** and **alter database** *appname.dbname* **set implicit_commit after** *n* **rows** • ESSCMD: SETDBSTATEITEM 22
Wait for write access to locked data block	"Understanding Isolation Levels" on page 1054	• Administration Services: Database Properties window, Transactions tab • MaxL: **alter database** *appname.dbname* **set lock_timeout** • ESSCMD: SETDBSTATEITEM 20
Pre-image access	"Understanding Isolation Levels" on page 1054	• Administration Services: Database Properties window, Transactions tab • MaxL: **alter database** *appname.dbname* **enable pre_image_access** • ESSCMD: SETDBSTATEITEM 19

Miscellaneous Database Settings

The following table describes miscellaneous database settings and lists the location of the settings in Administration Services, MaxL, and ESSCMD.

Table 70: Miscellaneous Database Settings

Setting	More Information	Location in Administration Services, MaxL, ESSCMD
Retrieval buffer size	"Setting the Retrieval Buffer Size" on page 1244	• Administration Services: Database Properties window, General tab • MaxL: **alter database** *appname.dbname* **set retrieve_buffer_size** *n* • ESSCMD: SETDBSTATEITEM 16
Retrieval sort buffer size	"Setting the Retrieval Sort Buffer Size" on page 1245	• Administration Services: Database Properties window, General tab • MaxL: **alter database** *appname.dbname* **set retrieve_sort_ buffer_size** *n* • ESSCMD: SETDBSTATEITEM 17
Data compression	"Data Compression" on page 1044	• Administration Services: Database Properties window, Storage tab • MaxL: **alter database** *appname.dbname* **enable compression** and **alter database** *appname.dbname* set **compression** *type* • ESSCMD: SETDBSTATEITEM 14 SETDBSTATEITEM 15
Maximum memory for trigger definitions	"Understanding Triggers Definitions" on page 130	MaxL: **create or replace trigger, alter trigger display trigger,** and **drop trigger**

For more information about these settings in Administration Services, see “Setting Database Properties” in the *Essbase Administration Services Online Help*.

Eliminating Fragmentation

Fragmentation is unused disk space. Fragmentation is created when Analytic Services writes a data block to a new location on disk and leaves unused space in the former location of the data block. Block size increases because data from a data load or calculation is appended to the blocks; the blocks must therefore be written to the end of a data file.

The Analytic Services Kernel merges adjacent fragments into increasingly larger fragments so that unused space is more likely to be re-used.

In some cases, fragmentation cannot be reduced completely. Fragmentation is likely to occur with the following:

- Read/write databases that users are constantly updating with data
- Databases that execute calculations around the clock
- Databases that frequently update and recalculate dense members
- Data loads that are poorly designed
- Databases that contain a significant number of Dynamic Calc and Store members
- Databases that use an isolation level of uncommitted access with commit block set to zero

If you experience performance slow-downs, you can check to see if there is too much fragmentation of the database, and if there is, you can take steps to reduce the fragmentation:

- “Measuring Fragmentation” on page 1121
- “Preventing or Removing Fragmentation” on page 1122

Measuring Fragmentation

You can measure fragmentation using the average clustering ratio or average fragmentation quotient statistic:

- "Using the Average Fragmentation Quotient" on page 1121
- "Using the Average Clustering Ratio" on page 1121

Using the Average Fragmentation Quotient

In ESSCMD, look at the Average Fragmentation Quotient that is returned when you execute GETDBSTATS command. Use this table to evaluate whether or not the level of fragmentation is likely to be causing performance problems:

Database Size	Fragmentation Quotient Threshold
Small (up to 200 Mb)	60% or higher
Medium (up to 2 Gb)	40% or higher
Large (greater than 2Gb)	30% or higher

Any quotient above the high end of the range indicates that reducing fragmentation may help performance, with the following qualifications:

- The reported value of the Fragmentation Quotient is more accurate when there are no other write transactions running on the database.
- For databases less than 50Mb using the Direct I/O access mode, the fragmentation quotient tends to be high. A high fragmentation quotient does not necessarily indicate a need to reduce fragmentation, because the free space is created in 8Mb chunks and all of it might not get used right away.

Using the Average Clustering Ratio

The average clustering ratio database statistic indicates the fragmentation level of the data (`.pag`) files. The maximum value, 1, indicates no fragmentation.

➤ To view the average clustering ratio for a database, use either of the following methods.

Tool	Topic	Location
Administration Services	Viewing Fragmentation Statistics	*Essbase Administration Services Online Help*
ESSCMD	GETDBSTATS	*Technical Reference*

Preventing or Removing Fragmentation

You can prevent and remove fragmentation:

- To prevent fragmentation, optimize data loads by sorting load records based upon sparse dimension members. For a comprehensive discussion of optimizing data load by grouping sparse members, see "Grouping Sparse Member Combinations" on page 1162.
- To remove fragmentation, perform an export of the database, delete all data in the database with CLEARDATA, and reload the export file. For discussion and instructions, see "Export Backups" on page 1082.
- To remove fragmentation, force a dense restructure of the database. See "Types of Database Restructuring" on page 1148.

Enabling Windows 4 GB RAM Tuning

Essbase supports Microsoft 4 GB RAM Tuning (4GT). 4GT enables users with extremely large databases to take advantage of a larger address space to improve performance.

The total addressable limit of RAM on servers running Windows 2000 or Windows NT is 4 GB. By default, applications can access 2 GB, with the Windows kernel using the other 2 GB. For selected versions of Windows running on Intel architecture servers, Microsoft provides the 4GT feature. The 4GT feature increases the addressable limit for applications to 3 GB, reducing the potential RAM allocated to the Windows kernel from 2 GB to 1 GB.

Essbase currently supports the 4GT feature on computers that use Intel-based processors with more than 2 GB of physical RAM, and that have either of the following versions of Windows:

- Windows 2000 Advanced Server
- Windows NT Server 4.0, Enterprise Edition

Enabling the Windows 4GT feature may benefit users if the Essbase installation has both of the following characteristics:

- Essbase is configured to use direct I/O. Enabling 4GT on Essbase installations configured for buffered I/O is not recommended.
- The index and data caches are sized correctly and Essbase performance is consistently improved by increasing the data file cache, but further increases are bounded by the previous 2 GB addressability limitation. For information about setting cache values, see Chapter 51, "Optimizing Analytic Services Caches."

Enabling 4GT

To configure the computer where Essbase is installed to enable the 4GT feature, modify the `boot.ini` file by adding `/3GB` to the startup line for each boot partition that is defined for a Windows version that supports 4GT; for example:

```
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(2)\WIN2KADV
[operating systems]
multi(0)disk(0)rdisk(0)partition(2)\WIN2KADV="Microsoft Windows
2000 Advanced Server" /3GB
multi(0)disk(0)rdisk(0)partition(2)\WINNT="Windows NT Server
Version 4.0" /3GB
```

Note:

This change to the `boot.ini` file is effective only if you are using Windows NT Server 4.0 Enterprise Edition or Windows 2000 Advanced Server. On standard Windows NT, although the /3GB flag relocates the kernel, applications are unable to access more than 2 GB.

On a dual boot system, to ensure that the boot loader supporting the 4GT feature is installed, be sure that the last operating system you install is a version that supports the 4GT feature.

Because the Windows kernel area is reduced, it is unlikely but conceivable that certain applications and workloads in certain environments may experience degraded performance. Testing your workload in your environment is recommended.

For additional information about the Microsoft Windows 4GT feature, see `www.microsoft.com`.

Finding Additional Optimization Information

"Using Database Settings to Customize for Maximum Performance" on page 1114 provides general-purpose information and does not account for the wide variety of configuration possibilities. For more information about performance and server, application, or other settings, see these chapters:

- For information on optimizing performance for a particular function, see these chapters:
  - Chapter 53, "Optimizing Data Loads"
  - Chapter 54, "Optimizing Calculations"
  - Chapter 55, "Optimizing with Intelligent Calculation"
  - Chapter 56, "Optimizing Reports and Other Types of Retrieval"
- For detailed information on the Analytic Services Kernel, see the following chapters:
  - Chapter 44, "Managing Database Settings"
  - Chapter 45, "Allocating Storage and Compressing Data"
  - Chapter 46, "Ensuring Data Integrity"
  - Chapter 51, "Optimizing Analytic Services Caches"
  - Appendix C, "Estimating Disk and Memory Requirements"

Chapter 51

Optimizing Analytic Services Caches

This chapter describes the memory caches that Analytic Services uses and provides recommendations for cache-related settings.

This chapter includes the following sections:

The caches described in this chapter are not relevant to aggregate storage databases. For information about the aggregate storage cache, see "Managing the Aggregate Storage Cache" on page 1345.

Understanding Analytic Services Caches

Analytic Services uses five memory caches to coordinate memory usage:

Table 71: Analytic Services Caches

Cache	Description
Index cache	The index cache is a buffer in memory that holds index pages. How many index pages are in memory at one time depends upon the amount of memory allocated to the cache.
Data file cache	The data file cache is a buffer in memory that holds compressed data files (`.pag` files). Analytic Services allocates memory to the data file cache during data load, calculation, and retrieval operations, as needed. The data file cache is used only when direct I/O is in effect.
Data cache	The data cache is a buffer in memory that holds uncompressed data blocks. Analytic Services allocates memory to the data cache during data load, calculation, and retrieval operations, as needed.
Calculator cache	The calculator cache is a buffer in memory that Analytic Services uses to create and track data blocks during calculation operations.
Dynamic calculator cache	The dynamic calculator cache is a buffer in memory that Analytic Services uses to store all of the blocks needed for a calculation of a Dynamic Calc member in a dense dimension (for example, for a query).

Analytic Services provides default size settings for each cache. You can adjust the size of any of these five caches as needed for each database. Appropriate cache size is affected by many factors, including database size, block size, index size, and available memory on the server. Cache size settings can effect database and general server performance significantly.

Use these topics for information and instructions about sizing caches for performance:

- “Deciding Whether to Use Cache Memory Locking” on page 1127
- “Sizing Caches” on page 1128
- “Fine Tuning Cache Settings” on page 1144

Deciding Whether to Use Cache Memory Locking

Before setting cache sizes, you need to enable cache memory locking or leave cache memory locking disabled (the default).

The setting for cache memory locking controls whether the memory used for the index cache, data file cache, and data cache is locked into physical memory, giving the Analytic Services kernel priority use of system RAM.

To use cache memory locking, you must be using direct I/O (buffered I/O is the default I/O access mode), and direct I/O requires a larger index cache size than buffered I/O. For more information, see "Migrating and Upgrading Databases" in the *Essbase Analytic Services Installation Guide*, and Chapter 44, "Managing Database Settings."

Locking improves performance for an Analytic Services database because the system memory manager does not need to swap the memory used by the caches when swapping the memory used by Analytic Server. By default, cache memory locking is turned off.

Enabling cache memory locking gives the Analytic Services Kernel priority use of system RAM. If you enable cache memory locking, leave at least one-third of the system RAM available for non-Analytic Services Kernel use. If you do not want to give the Analytic Services Kernel priority usage of system RAM, do not enable cache memory locking.

If you are running Analytic Services on Solaris, run the Bourne shell script, root.sh. before starting Analytic Services and enabling cache memory locking. This script sets the server to run in Superuser mode so that it can lock memory. For information about running the `root.sh` script, see the *Essbase Analytic Services Installation Guide*.

➤ To enable cache memory locking, use either of the following methods:

Tool	Topic	Location
Administration Services	Enabling Cache Memory Locking	*Essbase Administration Services Online Help*
MaxL	**alter database enable cache_pinning**	*Technical Reference*
ESSCMD	SETDBSTATEITEM 26	*Technical Reference*

Sizing Caches

The settings that you should use for each of the caches that you can configure depend on data distribution and the dense/sparse configuration of the database.

If memory resources are restricted, you can optimize performance by adjusting the cache settings relative to the memory available on the machine which contains your database.

The needs for each site and even for a particular database can vary. Depending on the complexity and type of each operation, Analytic Services allocates as much memory for the data file cache and the data cache as needed. Use the recommended values in this section to estimate enough memory for *optimal* performance.

If you are using Analytic Services for the first time, cache sizes are automatically set to the default values discussed in the following sections. If you are migrating from Analytic Services Release 5.x, the data file cache is set to the default value and the other cache settings from that version are retained when you migrate. See the *Essbase Analytic Services Installation Guide* for migration information.

Note: Changes made to cache sizes take effect the next time you start the database.

Use these topics to find and understand recommendations for each cache size:

- “Sizing the Index Cache” on page 1129
- “Changing the Index Cache Size” on page 1130
- “Sizing the Data Cache” on page 1132

- “Changing the Data Cache Size” on page 1133
- “Sizing the Calculator Cache” on page 1133
- “Sizing Dynamic Calculator Caches” on page 1141

Note: The size of index pages is fixed at 8 K. This is to reduce input-output overhead, as well as to simplify database migration.

Sizing the Index Cache

The index is stored in index files on disk. When a database is active, the most recently accessed index pages are held in the index cache, which is a memory area that is allocated for index pages. How much of the index can be held in memory at one time depends upon the amount of memory you allocate to the index cache.

When a data block is requested, Analytic Services looks at the index pages in the index cache to find the block location on disk. If the block location is not found in index pages in the index cache, the index page containing the block location is pulled into the index cache from disk. If the index cache is full, the least recently used index page in the cache is dropped to make room for the index page containing the location of the data block.

The effectiveness of the index cache size depends on the nature of the calculation you are performing. For example, if you were reloading and recalculating an entire database (such as a database that is refreshed each month), a high index cache size is not helpful because Analytic Services is creating new blocks rather than searching the index cache for existing blocks during calculation.

Table 72 shows default and recommended settings for the index cache.

Table 72: Index Cache Size Settings

Minimum Value	Default Value	Recommended Value
1024 KB (1048576 bytes)	Buffered I/O: 1024 KB (1048576 bytes) Direct I/O: 10240 KB (10485760 bytes)	Combined size of all `essn.ind` files, if possible; as large as possible otherwise. Do not set this cache size higher than the total index size, as no performance improvement results. To determine the total index size, see “Index Files” on page 1368.

For information about changing the I/O access mode for a database, or about changing the default for all newly created databases, see "Understanding Buffered I/O and Direct I/O" on page 1023.

In general, if you are using direct I/O, make the index cache as large as system resources allow, up to 2 GB. If you are using buffered I/O, make the index cache as small as possible.

For information and instructions on testing and fine tuning cache settings, see "Fine Tuning Cache Settings" on page 1144.

Changing the Index Cache Size

➤ To set the size of the index cache, use any of the following methods:

Tool	Topic	Location
Administration Services	Setting Cache Sizes	*Essbase Administration Services Online Help*
MaxL	**alter database set index_cache_size**	*Technical Reference*
ESSCMD	SETDBSTATEITEM 12 SETDBSTATE	*Technical Reference*

Sizing the Data File Cache

The data file cache holds data files (`.pag` files) in memory, if you are using direct I/O. If you are not using direct I/O, the data file cache is not used. How much of the data within data files can fit into memory at one time depends on the amount of memory you allocate to the data file cache.

In general, if you have to choose whether to allocate memory to the data cache or to the data file cache, choose the data file cache if you are using direct I/O.

Table 73 shows default and recommended settings for the data file cache.

Table 73: Data File Cache Size Settings

Minimum Value	Default Value	Recommended Value
Direct I/O: 10240 KB (10485760 bytes)	Direct I/O: 32768 KB (33554432 bytes)	Combined size of all `essn.pag` files, if possible; otherwise as large as possible. This cache setting not used if Analytic Services is set to use buffered I/O.

In general, if you are using direct I/O, make the data file cache as large as system resources allow, up to 2 GB. If you are using buffered I/O, the data file cache is not used.

For information and instructions on testing and fine tuning cache settings, see "Fine Tuning Cache Settings" on page 1144.

Changing the Data File Cache Size

➤ To set the size of the data file cache, use any of the following methods:

Tool	Topic	Location
Administration Services	Setting Cache Sizes	*Essbase Administration Services Online Help*
MaxL	**alter database set data_file_cache_size**	*Technical Reference*
ESSCMD	SETDBSTATEITEM 27	*Technical Reference*

Sizing the Data Cache

Data blocks reside on physical disk and in memory. The data cache is the memory area that is allocated to hold uncompressed data blocks. The number of blocks that can be held in the data cache at one time depends on the amount of memory you allocate to the data cache.

When a block is requested, Analytic Services searches the data cache for the block. If Analytic Services finds the block in the cache, it is accessed immediately. If the block is not found in the cache, Analytic Services searches the index for the appropriate block number and then uses the index entry of the block to retrieve it from the appropriate data file on disk. Retrieving a requested block from the data cache is faster, and therefore improves performance.

In general, if you have to choose whether to allocate memory to the data cache or to the data file cache, choose the data file cache if you are using direct I/O.

This table shows default and recommended settings for the data cache.

Table 74: Data Cache Size Settings

Minimum Value	Default Value	Recommended Value
3072 KB (3145728 bytes)	3072 KB (3145728 bytes)	0.125 * the value of data file cache size. Increase value if any of these conditions exist: • Many concurrent users are accessing different data blocks. • Calculation scripts contain functions on sparse ranges, and the functions require all members of a range to be in memory (for example, when using @RANK and @RANGE) • For data load, the number of threads specified by the DLTHREADSWRITE setting is very high and the expanded block size is large.

Make the data cache as small as possible whether you are using buffered I/O or direct I/O.

For information and instructions on testing and fine tuning cache settings, see "Fine Tuning Cache Settings" on page 1144.

Changing the Data Cache Size

➤ To set the size of the data cache, use any of the following methods:

Tool	Topic	Location
Administration Services	Setting Cache Sizes	*Essbase Administration Services Online Help*
MaxL	**alter database set data_cache_size**	*Technical Reference*
ESSCMD	SETDBSTATEITEM 5 SETDBSTATE	*Technical Reference*

Sizing the Calculator Cache

Analytic Services can create a bitmap, whose size is controlled by the size of the calculator cache, to record and track data blocks during a calculation. Determining which blocks exist using the bitmap is faster than accessing the disk to obtain the information, particularly if calculating a database for the first time or calculating a database when the data is very sparse.

Analytic Services uses the calculator cache bitmap if the database has at least two sparse dimensions, and either of these conditions are also met:

- You calculate at least one, full sparse dimension
- You specify the SET CACHE ALL command in a calculation script.

The best size for the calculator cache depends on the number and density of the sparse dimensions in your outline. Use these topics to understand the calculator cache bitmap, size the calculator cache, and change the size of the calculator cache (and therefore the largest possible size for the bitmap), if required:

- “Understanding the Calculator Cache Bitmap” on page 1134
- “Calculating the Calculator Cache Size” on page 1136
- “Changing the Calculator Cache with Calculation Scripts” on page 1140
- “Sizing the Calculator Cache to Calculate the Database for the First Time” on page 1140

Understanding the Calculator Cache Bitmap

For the calculator cache, Analytic Services separates sparse dimensions in the database into two groups:

- Bitmap dimensions: the sparse dimensions from the database outline that Analytic Services fits into the bitmap until the bitmap is full. Each member combination of the sparse dimensions placed in the bitmap occupies 1 bit of memory, and there must be enough space in the bitmap for every member combination of a sparse dimension for it to be placed in the bitmap.
- Anchoring dimensions: the remaining one or more sparse dimensions in the database outline that do not fit into the bitmap.

Analytic Services starts with the first sparse dimension in the database outline and fits as many sparse dimensions as possible into the bitmap. The dimensions that fit are the bitmap dimensions. Analytic Services stops the process when it cannot fit another complete sparse dimension into the bitmap. Because the calculator cache controls the size of the bitmap, the number of sparse dimensions that can fit in the bitmap depends on the size of the calculator cache (and the number and size of the sparse dimensions).

The remaining sparse dimensions are the anchoring dimensions. For anchoring dimensions, Analytic Services cannot use the bitmap to determine whether or not blocks exist.

To see which dimensions are anchoring dimensions and which are bitmap dimensions, use the SET MSG DETAIL calculation command to display bitmap information in the application log.

Carefully order the sparse dimensions in your outline so that as many dimensions as possible can be placed into the bitmap. Start with the dimension that contains the fewest members, and continue until the dimension with the most members is last. This order allows more dimensions to fit into the bitmap and results in improved calculation performance.

Note: The order of sparse dimensions in the outline also affects query performance. To optimize the outline for query performance, see “Optimizing Query Performance” on page 100.

Analytic Services uses a single bitmap if there is more than one anchoring dimension or if the calculator cache is not large enough to support multiple bitmaps, and uses two or more bitmaps if there is a single anchoring dimension.

A single bitmap has these properties:

- A single bitmap is used to track child blocks.
- A single bitmap uses the least memory but is less efficient than multiple bitmaps.

Multiple bitmaps have these properties:

- Two or more bitmaps are used, one to track child blocks and one to track parent blocks.
- Multiple bitmaps use more memory but are faster than using a single bitmap. The performance improvement is particularly high when you are calculating the database for the first time.
- The number of bitmaps used is determined by the maximum number of dependent parents for any of the members in the anchoring dimension. A member has one dependent parent, unless it has a shared member. For example, consider the Product dimension of the Sample Basic database. The member Cola (100-10) has one parent, which is Colas (100). However, Diet Cola (100-20) has two parents, which are Diet Drinks (Diet) and Colas (100). No members of Product have more than two dependent parents. Therefore, if Product is the anchoring dimension, the maximum number of dependent parents is 2.

Analytic Services chooses one of three options for the calculation:

Table 75: Options for calculator cache

Option	Method	Performance Rating
1	Single anchoring dimension, multiple bitmaps	1
2	Single anchoring dimension, single bitmap	2
3	Multiple anchoring dimensions, single bitmap	3

Analytic Services chooses the optimal performance method for a database calculation, based on the size of the calculator cache. If the calculator cache size is too small for any of the above options, Analytic Services does not use a calculator cache. Calculation performance may be significantly impaired.

Enabling parallel calculation may change which calculator cache option is used. See "Calculator Cache" on page 1186 for details.

CAUTION: If you are calculating the database for the first time, the size of the calculator cache is particularly significant for calculation performance. If possible, ensure that the calculator cache is large enough for Analytic Services to use the optimal calculator cache option.

Calculating the Calculator Cache Size

The optimum size of the calculator cache depends on the amount of memory the system has available. It also depends on the nature and configuration of the database.

Using the following formula, you can calculate the calculator cache size required for Analytic Services to choose each of the three options in Table 75 on page 1135.

Calculator cache	=	**Bitmap size in bytes * Number of bitmaps**
Bitmap size in bytes	=	Max ((member combinations on the bitmap dimensions/8), 4)
Number of bitmaps	=	Maximum number of dependent parents in the anchoring dimension + 2 constant bitmaps

Note: The minimum bitmap size is 4 bytes. If (member combinations on the bitmap dimensions/8) is less than 4 bytes, Analytic Services uses a bitmap size of 4 bytes.

Consider an example database with five sparse dimensions (S1 to S5):

Sparse Dimension	Number of Members	Dependent Parents
S1	20	Not applicable
S2	20	Not applicable
S3	50	Not applicable
S4	50	Not applicable
S5	200	3

Use this example information for these sample calculations:

- “Option 1: Single Anchoring Dimension, Multiple Bitmaps” on page 1137
- “Option 2: Single Anchoring Dimension, Single Bitmap” on page 1138
- “Option 3: Multiple Anchoring Dimensions, Single Bitmap” on page 1139

Option 1: Single Anchoring Dimension, Multiple Bitmaps

For this example calculation, assume the following facts about a database (from Table 75 on page 1135):

- Bitmap dimensions: S1, S2, S3, S4
- Anchoring dimension: S5
- Dependent parents in anchoring dimension: 3

Now perform this calculation:

Bitmap size in bytes	=	(S1 * S2 * S3 * S4) / 8
	=	(20 * 20 * 50 * 50) / 8
	=	125,000 bytes
Number of bitmaps	=	Maximum number of dependent parents in the anchoring dimension + 2 constant bitmaps
	=	3 + 2
	=	5
Calculator cache	=	Bitmap size * Number of bitmaps
	=	125,000 * 5
	=	**625,000 bytes**

In order for Analytic Services to use multiple bitmaps for this database with a single anchoring dimension, the calculator cache needs to be 625,000 bytes.

Option 2: Single Anchoring Dimension, Single Bitmap

For this example calculation, assume the following facts about a database (from Table 75 on page 1135):

- Bitmap dimensions: S1, S2, S3, S4
- Anchoring dimension: S5
- Dependent parents in anchoring dimension: Not applicable

Now perform this calculation:

Bitmap size in bytes	=	(S1 * S2 * S3 * S4) / 8
	=	(20 * 20 * 50 * 50) / 8
	=	125,000 bytes
Number of bitmaps	=	Single bitmap
	=	1
Calculator cache	=	Bitmap size * Number of bitmaps
	=	125,000 * 1
	=	**125,000 bytes**

In order for Analytic Services to use a single bitmap for this database with a single anchoring dimension, the calculator cache needs to be 125,000 bytes.

Option 3: Multiple Anchoring Dimensions, Single Bitmap

For this example calculation, assume the following facts about a database (from Table 75 on page 1135):

- Bitmap dimensions: S1, S2, S3
- Anchoring dimensions: S4, S5
- Dependent parents in anchoring dimensions: Not applicable

Now perform this calculation:

Bitmap size in bytes	=	(S1 * S2 * S3) / 8
	=	(20 * 20 * 50) / 8
	=	2,500 bytes
Number of bitmaps	=	Single bitmap
	=	1
Calculator cache	=	Bitmap size * Number of bitmaps
	=	2,500 * 1
	=	**2,500 bytes**

In order for Analytic Services to use a single bitmap for this database with multiple anchoring dimensions, the calculator cache needs to be 2,500 bytes.

Choosing a Calculator Cache Size for a Database

The following table shows which calculator cache option Analytic Services uses, depending on the calculator cache size specified:

Minimum Size Specified	Option Selected
625,000 bytes	Option 1 (provides optimal performance)
125,000 bytes	Option 2
2,500 bytes	Option 3

If you specify a calculator cache size of less than 2,500 bytes, Analytic Services does not use a calculator cache during the calculation. Calculation performance may be significantly impaired.

You can check which calculator cache option Analytic Services is able to use on a database by using the SET MSG SUMMARY command in a calculation script. Run the following calculation script on the empty database:

```
SET MSG SUMMARY;
CALC ALL;
```

Analytic Services displays the calculator cache setting in the ESSCMD window or in the application log. For a discussion of why you use SET MSG SUMMARY, see "SET MSG SUMMARY and SET MSG DETAIL" on page 1175.

The maximum calculator cache size that you can specify is 200,000,000 bytes. The default is 200,000 bytes. The calculator cache size that you choose depends on how much memory is available and the configuration of the database.

Note: The sizes of the calculator, index, data file, and data caches usually have a greater effect on performance if the database calculation is based more on aggregations and less on formula calculations.

Sizing the Calculator Cache to Calculate the Database for the First Time

If you are calculating the database for the first time, the size of the calculator cache is particularly significant. If possible, ensure that the calculator cache is large enough for Analytic Services to use the optimal calculator cache option. For discussion and examples of how to determine cache size, see "Calculating the Calculator Cache Size" on page 1136.

Changing the Calculator Cache with Calculation Scripts

You can use the default calculator cache size, or you can set the size of the calculator cache within a calculation script. If you set the size from a calculation script, the setting is used only for the duration of the calculation script. For details, review information about the calculation script SET CACHE command and the CALCCACHE configuration setting in the *Technical Reference*.

Sizing Dynamic Calculator Caches

Essbase uses a separate dynamic calculator cache for each open database. The DYNCALCCACHEMAXSIZE setting in the `essbase.cfg` file, specifies the maximum size of each dynamic calculator cache on the server. By default, the maximum size is 20 MB. Essbase allocates area in a dynamic calculator cache for data blocks until the maximum memory area specified by the DYNCALCACHEMAXSIZE setting is allocated.

For detailed information about DYNCALCACHEMAXSIZE and other dynamic calculator cache settings, see "Changing the Dynamic Calculator Cache Size" on page 1141.

Reviewing Dynamic Calculator Cache Usage

For each database, Analytic Services writes two messages to the application log for each data retrieval:

```
[Thu Oct 17 11:37:17 2002]Local/Sample///Info(1007125)
The number of Dynamic Calc Non-Store Members = [7 6 0 0 2 ]

[Thu Oct 17 11:37:17 2002]Local/Sample///Info(1007126)
The number of Dynamic Calc Store Members = [0 0 0 0 0 ]
```

The first message describes the total amount of time required for the retrieval. If a dynamic calculator cache is used, the second message displays the number of blocks calculated within the data calculator cache (DCC = *n*) and the number of blocks calculated in general memory (non-DCC = *n*).

Changing the Dynamic Calculator Cache Size

Five configuration file settings are relevant to dynamic calculator caches. The optimum values for these dynamic calculator cache settings depend on the amount of memory on the server machine, the configuration of all databases on the server machine, and the nature of user queries.

Table 76 describes each setting and includes recommendations on how to determine values for your system. To match your site's unique requirements, you may need to test and adjust the settings.

Table 76: essbase.cfg Settings for Dynamic Calculator Caches

DYNCALCCACHEMAXSIZE	
Description	This setting specifies the maximum size Essbase can allocate to each dynamic calculator cache on the server.
Recommended Setting	Recommended setting value = C * S * U. • C is the value of the appropriate CALCLOCKBLOCK setting in the `essbase.cfg` file. (The SET LOCKBLOCK command specifies which CALCLOCKBLOCK setting to use.) • S is the size of the largest expanded block across all databases on the machine. To calculate the expanded block size, multiply the number of members (including Dynamic Calc members) in each dense dimension together, and then multiply the result by the size of each member cell, 8 bytes. For example, for the dense dimensions in Sample Basic, 12 (Year) * 8 (Measures) * 3 (Scenario) * 8 bytes = 2304 bytes. • U is the maximum number of expected concurrent users on the database that has the largest number of concurrent users. Assigning the value 0 (zero) to DYNCALCACHEMAXSIZE tells Essbase not to use dynamic calculator caches. By default, the maximum size for this value is 20 MB (20,971,520 bytes).

Table 76: essbase.cfg Settings for Dynamic Calculator Caches (Continued)

DYNCALCCACHEWAITFORBLK	
Description	If Analytic Services uses all of the area allocated for a dynamic calculator cache, this setting tells Analytic Services whether to wait until space becomes available in the cache or to immediately write and calculate the blocks in memory outside the dynamic calculator cache. If the dynamic calculator cache is too small, it is possible for more than one thread to be in queue, each thread waiting to calculate its data blocks.
Recommended Setting	Recommended setting value = FALSE (default value). Before setting to TRUE, try these alternatives: • Add physical memory to the server machine • Increase the value of DYNCALCCACHEMAXSIZE, test, and repeat until you verify that you cannot use any more memory for the dynamic calculator cache.
DYNCALCCACHEBLKTIMEOUT	
Description	If Analytic Services is to wait for available space in the dynamic calculator cache, this setting defines how long it waits.
Recommended Setting	Recommended setting value = WT / B. • WT is the maximum tolerable wait time for a query; for example, 5 seconds. • B is the total number of logical blocks accessed in the largest query. To determine the value of B, check the messages in the application log for the largest number of Dyn.Calc.Cache “Big Block Allocs” for a query, as shown in Figure 192 on page 561.
DYNCALCCACHEBLKRELEASE	
Description	If Analytic Services has waited the specified time and space is still not available in the dynamic calculator cache, this setting tells Analytic Services whether to write and calculate the blocks immediately outside the dynamic calculator cache or to create space in the dynamic calculator cache by swapping out blocks and temporarily compressing the swapped blocks in a dynamic calculator cache compressed-block buffer.
Recommended Setting	Recommended setting value = FALSE (default value). Set to TRUE only if you are experiencing severe memory shortage problems.

Table 76: essbase.cfg Settings for Dynamic Calculator Caches (Continued)

DYNCALCCACHECOMPRBLKBUFSIZE	
Description	If Analytic Services has waited the specified wait time and the DYNCALCCACHEBLKRELEASE setting is TRUE, this setting is the size of the dynamic calculator cache compressed-block buffer.
Recommended Setting	Recommended setting value = (C * S) / 2. • C is the value of the current CALCLOCKBLOCK setting in the essbase.cfg file. The SET LOCKBLOCK command specifies which CALCLOCKBLOCK configuration setting is current. • S is the size of the largest expanded block across all databases on the machine. Calculate S as described for the DYNCALCCACHEMAXSIZE setting.

Note: After changing any parameter in the `essbase.cfg` file, you must stop and restart Analytic Server to use the new values.

For detailed information about specific dynamic calculator cache settings, see the *Technical Reference*.

Fine Tuning Cache Settings

After using a database at your site with typical data, user access, and standard environment (including server machines, network, etc.), check to see how Analytic Services performs. It is difficult to predict optimal cache sizes without testing. You may need to adjust cache settings.

Understanding Cache Settings

The sizes of the index cache and the data file cache (when direct I/O is used) are the most critical Analytic Services cache settings. In general, the larger these caches, the less swapping activity occurs; however, it does not always help performance to set cache sizes larger and larger. Read this entire section to understand cache size considerations.

Index Cache

The advantages of a large index cache start to level off after a certain point. Whenever the index cache size equals or exceeds the index size (including all index files on all volumes), performance does not improve. However, to account for future growth of the index, you can set the index cache size larger than the current index size. Because the index cache is filled with index pages, for optimum use of storage, set the size of the index cache to be a multiple of the size of the index page (8 KB). See "Index Files" on page 1368 for an example of estimating index size.

Data File Cache

If possible, set the data file cache to equal the size of the stored data, which is the combined size of all `ess*.pag` files. Otherwise, the data file cache should be as large as possible. If you want to account for future growth of stored data, you can set the data file cache size larger than the current size of stored data.

Note: The data file cache is used only if you are using direct I/O.

Data Cache

The data cache should be about 0.125 times the data file cache. However, certain calculations require a larger data cache size. If many concurrent users are accessing different data blocks, this cache should be larger.

In general, if you have to choose between allocating memory to the data file cache or allocating it to the data cache, choose the data file cache if you are using direct I/O. If you are migrating from a previous version of Analytic Services, see the *Essbase Analytic Services Installation Guide* for relevant migration information.

Checking Cache Hit Ratios

Every cache has a "hit ratio." The hit ratio indicates the percentage of time that a requested piece of information is available in the cache. You can check the hit ratio of the index cache, the data cache, and the data file cache to determine whether you need to increase the cache size.

➤ To check cache hit ratios, see “Checking Cache Hit Ratios” in the *Essbase Administration Services Online Help*.

- The cache hit ratio indicates the percentage of time that a requested piece of information is already in the cache. A higher hit ratio indicates that the data is in the cache more often. This improves performance because the requested data does not have to be retrieved from disk for the next process. A hit ratio of 1.0 indicates that every time data is requested, it is found in the cache. This is the maximum performance possible from a cache setting.
- The Hit Ratio on Index Cache setting indicates the Analytic Services Kernel success rate in locating index information in the index cache without having to retrieve another index page from disk.
- The Hit Ratio on Data File Cache setting indicates the Analytic Services Kernel success rate in locating data file pages in the data file cache without having to retrieve the data file from disk.
- The Hit Ratio on Data Cache setting indicates the Analytic Services success rate in locating data blocks in the data cache without having to retrieve the block from the data file cache.

5. Check memory allocation. Add smaller amounts of memory at a time, if needed, because a smaller increment may have the same benefit as a large one. Large, incremental allocations of memory usually result in very little gain in the hit ratio.

Checking Performance

You can check cache statistics for a database by using the `GETPERFSTATS` command in ESSCMD.

Chapter 49, “Monitoring Performance,” provides detailed information about ways to check performance.

Running Test Calculations

Because calculations are the most processor-intensive operations on a Analytic Services database, you should run test calculations and examine how various cache sizes affect memory use on Analytic Server.

Chapter 52

Optimizing Database Restructuring

This chapter describes how changes to a database outline affect Analytic Services:

In addition to the information in this chapter, look for information and instructions concerning restructuring in the following topics:

The information in this chapter is not relevant to aggregate storage databases.

Database Restructuring

As your business changes, you change the Analytic Services database outline to capture new product lines, provide information on new scenarios, reflect new time periods, etc. Some changes to a database outline affect the data storage arrangement, forcing Analytic Services to restructure the database.

Because changes that require restructuring the database are very time-consuming, (unless you discard the data before restructuring), you may wish to make decisions about these kinds of changes, based on how much they affect performance. This

section provides the information you need to understand how restructuring affects performance, and describes tasks you can perform related to database restructuring:

Note: For information about clearing data, and thus avoiding some restructuring, see CLEARDATA and CLEARBLOCK in the *Technical Reference* or Clearing Data in the *Essbase Administration Services Online Help*.

Types of Database Restructuring

You can restructure a database explicitly or implicitly. To explicitly restructure a database, use the `alter database` *`DBS-NAME`* `force restructure` MaxL command. See "alter database" in the *Technical Reference* for more information.

An explicit restructure triggers a full restructure of the database.

Analytic Services uses three types of implicit restructure operations:

- Full restructure: If a member of a dense dimension is moved, deleted, or added, Analytic Services restructures the blocks in the data files and creates new data files. When Analytic Services restructures the data blocks, it regenerates the index automatically so that index entries point to the new data blocks. Analytic Services marks all restructured blocks as dirty, so after a full restructure you need to recalculate the database. Full restructuring is the most time-consuming of the restructures and, for large databases, can take a very long time to complete.
- Sparse restructure: If a member of a sparse dimension or a member of an attribute dimension is moved, deleted, or added, Analytic Services restructures the index and creates new index files. Restructuring the index is relatively fast; the amount of time required depends on the size of the index.
- Outline-only restructure: If a change affects only the database outline, Analytic Services does not restructure the index or data files. Member name changes, creation of aliases, and dynamic calculation formula changes are examples of changes that affect only the database outline.

If you use incremental restructuring, Analytic Services defers full restructuring. If you change a database outline frequently, consider enabling incremental restructuring. See "Incremental Restructuring and Performance" on page 1152 for a comprehensive discussion of incremental restructuring.

Note: How a database outline is changed (that is, by using Outline Editor or using dimension build) does not influence restructuring. Only the type of information change influences what type of restructuring, if any, takes place.

Conditions Affecting Database Restructuring

Intelligent Calculation, name changes, and formula changes affect database restructuring:

- If you use Intelligent Calculation in the database, all restructured blocks are marked as dirty whenever data blocks are restructured. Marking the blocks as dirty forces the next default Intelligent Calculation to be a full calculation.
- If you change a name or a formula, Analytic Services does not mark the affected blocks as dirty. Therefore, you must use a method other than full calculation to recalculate the member or the database.

Use this table to find information about restructuring:

Table 77: Topics Related To Database Restructuring

Topic	Related Information
Intelligent Calculation	"Restructuring Databases" on page 1241
Sparse and dense dimensions	• "Sparse and Dense Dimensions" on page 62, "Selection of Sparse and Dense Dimensions" on page 68 • "Dense and Sparse Selection Scenarios" on page 70
Attribute dimensions	"Designing Attribute Dimensions" on page 192
Dimension building	Chapter 16, "Understanding Data Loading and Dimension Building"
Outline Editor	Chapter 8, "Creating and Changing Database Outlines"

Temporary Files Used During Restructuring

When Analytic Services restructures both the data blocks and the index, it uses these files:

Table 78: Files Used During Database Restructuring

File	Description
ess*xxxxx*.pag	Analytic Services data file
ess*xxxxx*.ind	Analytic Services index file
dbname.esm	Analytic Services kernel file that contains control information used for database recovery
dbname.tct	Transaction control table
dbname.ind	Free fragment file for data and index free fragments
dbname.otl	Outline file that stores all metadata for a database and defines how data is stored. The outline file does not store data.

Full Restructures

To perform a full restructure, Analytic Services does the following:

1. Creates temporary files that are copies of the .ind, .pag, .otl, .esm, and .tct files. Each temporary file substitutes either N or U for the last character of the file extension, so the temporary file names are *dbname*.inn, ess*xxxxx*.inn, ess*xxxxx*.pan, *dbname*.otn, *dbname*.esn, and *dbname*.tcu.
2. Reads the blocks from the database files copied in step 1, restructures the blocks in memory, and then stores them in the new temporary files. This step takes the most time.
3. Removes the database files copied in step 1, including .ind, .pag, .otl, .esm, and .tct files.
4. Renames the temporary files to the correct file names: .ind, .pag, .otl, .esm, and .tct.

Sparse Restructures

When Analytic Services does a sparse restructure (restructures just the index), it uses the following files:

- `essxxxxx.ind`
- *`dbname`*`.otl`
- *`dbname`*`.esm`

To perform a sparse restructure, Analytic Services does the following:

1. Renames the *`dbame`*`.esm` file to *`dbname`*`.esr`
2. Renames the `ess`*`xxxxx`*`.ind` files to `ess`*`xxxxx`*`.inm`.
3. Creates new index files (`ess`*`xxxxx`*`.ind`) to store index information that is changed by the restructuring operation.
4. Removes *`dbname`*`.esr` and `ess`*`xxxxx`*`.inm` created in step 1.

Optimization of Restructure Operations

If a database outline changes frequently, analyze the outline and the types of changes that you are making. Remember that changes to sparse dimensions or attribute dimensions are relatively fast because only the index needs to change. Changes to dense dimensions are relatively slow because data blocks need to be rebuilt.

These types of restructure operations are listed from fastest to slowest:

- Outline only (no index or data files)
- Sparse (only index files)
- Full (index files and data files) as a result of adding, deleting, or moving members and other operations as listed in Table 79 on page 1155
- Full (index and data files) as a result of changing a dense dimension to sparse or changing a a sparse dimension to dense.

Actions That Improve Performance

There are a number of things you can do to improve performance related to database restructuring:

- If you change a dimension frequently, make it sparse.
- Use incremental restructuring to control when Analytic Services performs a required database restructuring.
- Select options when you save a modified outline that reduce the amount of restructuring required.

Incremental Restructuring and Performance

If you make frequent changes to a database outline, you may want to consider enabling incremental restructuring. When incremental restructuring is enabled, Analytic Services defers restructuring so that a change to the database outline or to a dimension does not cause structural change. Analytic Services restructures the index and, if necessary, the affected block the next time the block is accessed.

Understanding Incremental Restructuring

When incremental restructuring is enabled, Analytic Services defers restructuring for the database changes listed in Table 79 on page 1155, unless otherwise noted in the table.

The following changes override incremental restructuring; that is, they result in immediate restructuring, regardless of whether incremental restructuring is enabled:

- Adding or deleting a non-attribute dimension.
- Deleting a stored member of a sparse dimension.
- Changing a dimension definition from sparse to dense or from dense to sparse.
- If you are using linked reporting objects (LROs) in a database, incremental restructuring is automatically disabled on that database. Disabling of incremental restructuring does not affect other databases on the server.
- Certain member additions and certain changes to sparse dimensions can also trigger immediate restructuring. For detailed descriptions of the effects of various actions, see Table 79 on page 1155.

Regardless of whether incremental restructuring is enabled, if an outline has already been incrementally restructured (a full restructure is already pending), adding shared members causes Essbase to perform a full restructure.

Note: Recalculate the database after any type of restructure operation.

Using Incremental Restructuring

You can enable incremental restructuring for any of the following databases:

- An individual database in an application
- All databases in an application
- All databases in all applications

To enable incremental restructuring, use the INCRESTRUC setting in the `essbase.cfg` file. For detailed information on the INCRESTRUC setting and for syntax, see the *Technical Reference*.

Analytic Services logs outline changes in an internal file, *`dbname`*`.ocl`. Analytic Services clears the file when it does a full database restructure or when you clear or reset the database. The file *`dbname`*`.ocl` can grow quite large. To clear this file, issue VALIDATE in ESSCMD. VALIDATE causes Analytic Services to restructure any blocks whose restructure was deferred; thus, the file is cleared. When you issue VALIDATE, make sure that the database is *not* in read-only mode (read-only mode is used for backing up a database). For detailed information on the VALIDATE command, see “Using VALIDATE to Check Integrity” on page 1067.

Options for Saving a Modified Outline

Analytic Services displays a dialog box when you save outline changes that trigger database restructuring (using Outline Editor). In the Restructure Database dialog box, you define how data values should be handled during restructure; for example, you can choose to preserve all data, to preserve only level 0 or input data, or to discard all data during restructure. For more information, see “Saving Outlines” in *Essbase Administration Services Online Help*.

If the database contains data, you need enough free disk space on the server to create a backup copy of the database. Backup ensures that any abnormal termination during the restructure process does not corrupt the database.

Analytic Services may display a “Restructuring not required” message, yet still perform an index-only restructure. This event is most likely to occur if you make changes to a sparse dimension. If you try to cancel a restructure operation, Analytic Services may issue a “Can’t cancel” message. If such a message is displayed, Analytic Services is performing final cleanup and it is too late to cancel.

Outline Change Log

If you activate the outline change log, Analytic Services records all activity that affects the outline (member name changes, member moves, and so on). The more changes you make to the outline, the more updates Analytic Services must make to the log, thus slowing performance.

By default, Analytic Services does not log outline changes. To see if outline logging is slowing performance, look for OUTLINECHANGELOG TRUE in the `essbase.cfg` file. For a comprehensive discussion of the outline change log, see “Understanding and Using the Outline Change Log” on page 1002.

Analytic Services Partitioning Option

When you use Partitioning, Analytic Services tracks outline changes so that you can synchronize the database outlines across partitions. Tracking outline changes slows restructuring, particularly when there are many structural changes.

If Analytic Services restructures data when you are using partitioning, perform the following steps to make sure that data is synchronized across partitions:

1. Validate the partitions.

   For a brief discussion and instructions, see “Validating Partitions” on page 281.

   Note: To validate a partition, you must have database designer permissions or higher.

2. Synchronize the outlines of the partitions.

   For a comprehensive discussion and instructions, see “Synchronizing Outlines” on page 284.

Outline Change Quick Reference

Table 79 shows all outline changes that affect calculation and restructuring, including incremental restructuring.

Note: If you are using Partitioning, restructuring affects only the database to which you are connected.

Table 79: How Actions Affect Databases and Restructuring

Action	Calculation and Standard Restructure Effects	Incremental Restructuring Applies? (If Enabled)
Delete, Add, or Move Member		
Delete member of sparse dimension	Data needs to be recalculated to reflect changes to relationships. Analytic Services deletes from the index file all pointers to blocks represented by the deleted member. Because the blocks are no longer pointed to, they become free space.	For regular members, no. Analytic Services restructures the index, overriding incremental restructure. For label-only members, yes, restructuring is deferred.
Delete member of attribute dimension	None	No
Delete member of dense dimension	Data needs to be recalculated to reflect changes to relationships. Analytic Services restructures the data files to reflect a changed block size. Analytic Services restructures the index.	Yes. Restructure deferred.
Delete shared member in sparse or dense dimension	Data needs to be recalculated. The data remains associated with the original member name, but, because the parent of the shared member may have depended on the child data, recalculation is needed. No restructure.	No

Table 79: How Actions Affect Databases and Restructuring (Continued)

Action	Calculation and Standard Restructure Effects	Incremental Restructuring Applies? (If Enabled)
Add member to sparse dimension	Data for the new member needs to be loaded or calculated to derive new values. Analytic Services restructures the index.	Yes. Restructure deferred.
Add member to dense dimension	Data for the new member needs to be loaded or calculated to derive new values. Data needs to be recalculated. Analytic Services restructures the data files to reflect a changed block size. Analytic Services restructures the index.	Yes. Restructure deferred.
Add member to attribute dimension	None	No
Add shared member to sparse or dense dimension	Data needs to be recalculated. The new shared member affects the consolidation to its parent. No restructure.	No
Move regular member within a sparse dimension	Data needs to be recalculated to reflect changes in consolidation. Analytic Services restructures the index file.	No. Analytic Services restructures the index file, overriding incremental restructure.
Move regular member within a dense dimension	Data needs to be recalculated to reflect changes in consolidation. Analytic Services restructures both index and data files.	Yes. Restructure deferred.
Move an attribute dimension member	None	No

Table 79: How Actions Affect Databases and Restructuring (Continued)

Action	Calculation and Standard Restructure Effects	Incremental Restructuring Applies? (If Enabled)
Other Member-Related Changes		
Change a member alias or add an alias to a member	None	No
Rename member	None	No
Change member formula	Data needs to be recalculated to reflect formula changes. No restructure.	No
Dynamic Calculation-Related Changes		
Define Dynamic Calc member as Dynamic Calc and Store	For dense dimension members: Analytic Services restructures both index and data files. For sparse dimension members: no restructure.	Yes. Restructure deferred.
Define Dynamic Calc and Store member as Dynamic Calc	None	No
Define regular dense dimension member as Dynamic Calc and Store	None	No
Define regular dense dimension member as Dynamic Calc	Analytic Services restructures both index and data files.	Restructure deferred.
Define sparse dimension Dynamic Calc and Store member or Dynamic Calc member as regular member	No restructure.	No
Define sparse dimension regular member as Dynamic Calc or Dynamic Calc and Store	Analytic Services restructures both index and data files.	Yes. Restructure deferred.

Table 79: How Actions Affect Databases and Restructuring (Continued)

Action	Calculation and Standard Restructure Effects	Incremental Restructuring Applies? (If Enabled)
Define dense dimension Dynamic Calc and Store member as regular member	No restructure.	No
Define dense dimension Dynamic Calc member as regular member	Analytic Services restructures both index and data files.	Yes. Restructure deferred.
Define dense dimension regular member as Dynamic Calc member	Analytic Services restructures both index and data files.	Yes. Restructure deferred.
Add, delete, or move sparse dimension Dynamic Calc member	Analytic Services restructures only index files.	For member add or delete, restructure is deferred. For member move, Analytic Services restructures only index files, overriding incremental restructure.
Add, delete, or move sparse dimension Dynamic Calc and Store member	Analytic Services restructures only index files.	For member add, restructure deferred. For member move or delete, Analytic Services restructures only index files (overrides incremental restructure).
Add, delete, or move dense dimension Dynamic Calc and Store member	Analytic Services restructures both index and data files.	No
Add, delete, or move dense dimension Dynamic Calc member	No restructure.	No

Table 79: How Actions Affect Databases and Restructuring (Continued)

Action	Calculation and Standard Restructure Effects	Incremental Restructuring Applies? (If Enabled)
Property and Other Changes		
Change dense-sparse property	Data needs to be recalculated. Analytic Services restructures both index and data files.	Analytic Services restructures both index and data files overriding incremental restructure.
Change label only property	Data needs to be recalculated. Analytic Services restructures both index and data files.	Restructure deferred.
Change shared member property	Data needs to be recalculated to reflect the changed data value of the child. Analytic Services restructures both index and data files.	Restructure deferred.
Change properties other than dense-sparse, label, or shared	Data may need to be recalculated to reflect changed consolidation properties, such as changing time balance from first to last.	No
Change the order of two sparse dimensions	No calculation or data load impact. Analytic Services restructures the index.	Analytic Services restructures the index, overriding incremental restructure.
Change the order of dimensions	Data needs to be recalculated. Analytic Services restructures both index and data files.	Analytic Services restructures both index and data files (overrides incremental restructure).
Change the order of attribute dimensions	None	No
Create, delete, clear, rename, or copy an alias table	None	No
Import an alias table or set a member alias	None	No

Table 79: How Actions Affect Databases and Restructuring (Continued)

Action	Calculation and Standard Restructure Effects	Incremental Restructuring Applies? (If Enabled)
Change the case-sensitive setting	None	No
Name a level and generation	None	No
Create, change, or delete a user-defined attribute	None	No

Chapter

53

Optimizing Data Loads

Loading a large data source into an Analytic Services database can take hours. You can speed up the data loading process by improving two areas:

- Minimizing the time spent reading and parsing the data source
- Minimizing the time spent reading and writing to the database

This chapter contains the following sections:

Understanding Data Loads

This section is not relevant to aggregate storage databases.

To optimize data load performance, you must think in terms of database structure. Analytic Services loads data block by block. For each unique combination of sparse dimension members, one data block contains the data for all the dense dimension combinations, assuming there is at least one cell containing data. For faster access to block locations, Analytic Services uses an *index*. Each entry in the index corresponds to one data block. For further explanation of how sparse and

dense dimensions affect database structure, see "Sparse and Dense Dimensions" on page 62, "Selection of Sparse and Dense Dimensions" on page 68, and "Dense and Sparse Selection Scenarios" on page 70.

When Analytic Services loads a data source, Analytic Services processes the data in three main stages:

- Input stage: Analytic Services reads a portion of the data source.
- Preparation stage: Analytic Services arranges the data in preparation for putting it into blocks.
- Write stage: Analytic Services puts the data into blocks in memory and then writes the blocks to disk, finding the correct block on the disk by using the index, which is composed of pointers based on sparse intersections.

This process is repeated until all data is loaded. By using one or more processing threads in each stage, Analytic Services can perform some processes in parallel. For a description of the parallel data load process, see "Managing Parallel Data Load Processing" on page 1167.

All examples in this chapter assume that you are familiar with the discussions about data sources in "Data Sources" on page 356.

Grouping Sparse Member Combinations

This section is not relevant to aggregate storage databases.

The most effective strategy to improve performance is to minimize the number of disk I/Os that Analytic Services must perform while reading or writing to the database. Because Analytic Services loads data block by block, organizing the source data to correspond to the physical block organization reduces the number of physical disk I/Os that Analytic Services must perform.

Arrange the data source so that records with the same unique combination of sparse dimensions are grouped together. This arrangement corresponds to blocks in the database.

The examples in this chapter illustrate various ways you can organize the data following this strategy. These examples use a subset of the Sample Basic database, as shown in Table 80.

Table 80: Dimensions and Values for Examples

Sparse, Non attribute Dimensions	Dense Dimensions
Scenario (Budget, Actual)	Measures (Sales, Margin, COG, Profit)
Product (Cola, Root Beer)	Year (Jan, Feb)
Market (Florida, Ohio)	

Note: Because you do not load data into attribute dimensions, they are not relevant to this discussion even though they are sparse.

First, consider the data shown in Figure 238. Because it is not grouped by sparse-dimension member combinations, this data has not been sorted for optimization. As Analytic Services reads each record, it must deal with different members of the sparse dimensions.

Figure 238: Non-Optimized Sequence of Source Data

```
Jan
Actual    Cola          Ohio      Sales    25
Budget    "Root Beer"   Florida   Sales    28
Actual    "Root Beer"   Ohio      Sales    18
Budget    Cola          Florida   Sales    30
```

This data loads slowly because Analytic Services accesses four different blocks instead of one.

From the same Sample Basic database, Figure 239 shows different records sorted by a unique combination of sparse-dimension members: Actual -> Cola -> Ohio. Analytic Services accesses only one block to load these four records.

Figure 239: Optimally-Organized Source Data

```
Actual     Cola    Ohio    Jan   Sales     25
Actual     Cola    Ohio    Jan   Margin    18
Actual     Cola    Ohio    Jan   COGS      20
Actual     Cola    Ohio    Jan   Profit     5
```

You can use a data source that loads more than one cell per record. Make sure that records are grouped together by unique sparse-dimension member combinations, then order the records so that the dimension in the record for which you provide multiple values is a dense dimension.

Figure 240 uses a header record to identify the members of the Measures dimension, which is a dense dimension. The data is sorted first by members of the dense dimension Year and grouped hierarchically by members of the other dimensions. Multiple values for the Measures dimension are provided on each record.

Figure 240: Source Data Sorted and Grouped by Dense Dimensions

```
                                      Sales  Margin   COG  Profit
Jan Actual  Cola          Ohio          25      18    20       5
Jan Actual  Cola          Florida       30      19    20      10
Jan Actual  "Root Beer"   Ohio          18      12    10       8
Jan Actual  "Root Beer"   Florida       28      18    20       8
```

The heading and first data line in this example provide the same data shown in four lines in Figure 239.

For detailed information, including examples, about arranging data in source files before loading, see "Data Sources That Do Not Need a Rules File" on page 365.

Making the Data Source As Small As Possible

Make the data source as small as possible. The fewer fields that Analytic Services reads in the data source, the less time is needed to read and load the data.

Group the data into ranges. Eliminating redundancy in the data source reduces the number of fields that Analytic Services must read before loading data values.

Figure 241 shows a file that is not organized in ranges. It includes unneeded repetition of fields. All values are Profit values. Profit needs to be included only at the beginning of the group of data applicable to it. This example contains 33 fields that Analytic Services must read in order to load the data values properly.

Figure 241: Data Source Without Ranges

```
Profit
Jan     "New York"   Cola          4
Jan     "New York"   "Diet Cola"   3
Jan     Ohio         Cola          8
Jan     Ohio         "Diet Cola"   7
```

```
Feb     "New York"   Cola          6
Feb     "New York"   "Diet Cola"   8
Feb     Ohio         Cola          7
Feb     Ohio         "Diet Cola"   9
```

Figure 242 shows the same file optimized by grouping members in ranges. By eliminating redundancy, this example contains only 23 fields that Analytic Services must read in order to load the data values properly.

Figure 242: Data Source Organized in Ranges

```
Profit
Jan   "New York"   Cola          4
                   "Diet Cola"   3
      Ohio         Cola          8
                   "Diet Cola"   7
Feb   "New York"   Cola          6
                   "Diet Cola"   8
      Ohio         Cola          7
                   "Diet Cola"   9
```

Analytic Services assigns the first value, 4, to Jan->New York->Cola; it assigns the next value, 3, to Jan->New York->Diet Cola and so on.

Although sorted efficiently, the data in Figure 240 still shows a lot of repetition that can slow down the load process. You can further optimize this data by grouping the data into ranges. The optimized data source shown in Figure 243 eliminates the redundant fields, thereby reducing processing time.

Figure 243: Source Data Sorted and Grouped in Ranges

```
                                    Sales  Margin   COG  Profit
Jan Actual  Cola          Ohio        25      18    20       5
                           Florida     30      19    20      10
            "Root Beer"    Ohio        18      12    10       8
                           Florida     28      18    20       8
```

For information about and examples of organizing source data into ranges, see “Formatting Ranges of Member Fields” on page 367.

Making Source Fields As Small As Possible

Making fields in a data source smaller enables Analytic Services to read and load the data in less time.

Make the fields in the data source as small as possible by performing the following tasks:

- Removing excess white space in the data source. For example, use tabs instead of blank spaces.
- Rounding off computer-generated numbers to the precision you need. For example, if the data value has nine decimal points and you only care about two, round the number to two decimal points.
- Using #MI instead of #MISSING.

Positioning Data in the Same Order As the Outline

This section is not relevant to aggregate storage databases.

The index is organized in the same order as the sparse dimensions in the outline. To further optimize the data source, with the sparse data combinations in the data source grouped together, arrange the data so that sparse dimensions are in the same order as the outline.

Analytic Services pages portions of the index in and out of memory as requested by the data load or other operations. Arranging the source data to match the order of entries in the index speeds up the data load because it requires less paging of the index. Less paging results in fewer I/O operations.

Analytic Services uses the index cache size to determine how much of the index can be paged into memory. Adjusting the size of the index cache may also improve data load performance.

Note: If the index cache size is large enough to hold the entire index in memory, positioning data in the same order as the outline does not affect the speed of data loads.

For detailed information about setting the index cache size, see "Sizing the Index Cache" on page 1129.

Loading from Analytic Server

Loading the data source from the Analytic Server computer is faster than loading from a client computer. To load a data source from the server, move the data source to the server computer and then start the load.

Loading data from the server improves performance because the data does not have to be transported over the network from the client computer to the server computer.

Managing Parallel Data Load Processing

The methods described earlier in this chapter give you the most substantial data load performance enhancements. If you have not done so, you also need to carefully evaluate your processor speed and memory requirements and upgrade your computers to meet these requirements.

Another method to speed up data loads is to work with the Analytic Services parallel data load feature to optimize use of processor resources. The parallel data load feature recognizes opportunities to process data load tasks at the same time. Although some opportunities present themselves on single-processor computers, many more opportunities are available on multiple-processor computers.

To enable you to fine tune processor use for specific application and database situations, Analytic Services provides three `essbase.cfg` settings: DLTHREADSPREPARE, DLTHREADSWRITE, and DLSINGLETHREADPERSTAGE.

Understanding Parallel Data Load Processing

When Analytic Services loads a data source, it works with a portion of data at a time, in stages. Analytic Services looks at each stage as a task and uses separate processing *threads* in memory to perform each task.

One form of parallel processing occurs when one thread takes advantage of processor resources that are left idle during the wait time of another thread. For example, while a thread performs I/O processing, it must wait for the slower hardware to perform its task. While this thread waits, another thread can use the idle processor resource. Processing staged tasks in parallel can improve processor efficiency by minimizing idle time.

When computers have multiple processors, Analytic Services can perform an additional form of parallel processing. When a data load stage completes its work on a portion of data, it can pass the work to the next stage and start work immediately on another portion of data. Processing threads perform their tasks simultaneously on the different processors, providing even faster throughput.

Optimizing Parallel Data Load Processing

Even though Analytic Services uses parallel processing to optimize processor resources across the data load stages, there are still times when processor resources can be idle. To take advantage of these idle times, Analytic Services can further divide up record processing in the preparation and write stages. To tailor parallel processing to your situation, you can use the DLTHREADSPREPARE and DLTHREADSWRITE `essbase.cfg` settings to tell Analytic Services to use additional threads during these stages.

Setting Parallel Data Load Settings

As shown in Table 81, Analytic Services provides three `essbase.cfg` settings that enable you to manage parallel data load processing.

You can specify setting values that apply to all applications on a given Analytic Server or you can specify settings multiple times with different values for different applications and databases.

Table 81: Parallel Data Load essbase.cfg Settings

Setting	Description
DLTHREADSPREPARE	Specifies how many threads Analytic Services may use during the data load stage that codifies and organizes the data in preparation to being written to blocks in memory
DLTHREADSWRITE	Specifies how many threads Analytic Services may use during the data load stage that writes data to the disk. High values may require allocation of additional cache. For an explanation of the relationship between DLTHREADSWRITE and data cache size, see "Implications in Sizing the Data Cache" on page 1169. **Note:** For aggregate storage databases, Analytic Server uses one thread with aggregate storage cache. The DLTHREADSWRITE setting is ignored.

Table 81: Parallel Data Load essbase.cfg Settings (Continued)

Setting	Description
DLSINGLETHREADPERSTAGE	Specifies that Analytic Services use a single thread per stage, ignoring the values in the DLTHREADSPREPARE and DLTHREADSWRITE settings

Only when the DLSINGLETHREADPERSTAGE setting is set to FALSE for the specific application and database being loaded does the data load process use the thread values specified in the DLTHREADSPREPARE and DLTHREADSWRITE settings.

See the *Technical Reference* for details about these settings and their parameters.

Implications in Sizing the Data Cache

For block storage databases, Analytic Server allocates the data cache memory area to hold uncompressed data blocks. Each thread specified by the DLTHREADSWRITE setting uses an area in the data cache equal to the size of an expanded block.

Depending on the size of the block, the number of threads, and how much data cache is used by other concurrent operations during a data load, it may be possible to need more data cache than is available. In such circumstances, decrease the number of threads or increase the size of the data cache.

➤ To change the data cache size, see "Changing the Data Cache Size" on page 1133.

Testing Different Thread Values

While processing data loads, you can view processor activity. Different operating systems provide different tools for viewing processor activity. For example, the Task Manager in Windows/NT and Windows/2000 enables you to view processor and memory usage and processes. Among the tools available on UNIX are top and vmstat. You can also use third-party tools to view and analyze system utilization.

➤ To assess system usage during data load processing;

1. Start with the default parallel data load processing thread values whereby Analytic Services uses a single thread per stage.
2. Perform and time the data load.
3. Monitor the entire process, identifying the stages during which the processor may be idle.
4. Alter the `essbase.cfg` settings that are described in "Setting Parallel Data Load Settings" on page 1168.
5. Repeat the last three steps until you find values that provide the best performance.

Chapter 54

Optimizing Calculations

This chapter provides information on how to optimize the performance of Analytic Services calculations in block storage databases:

In addition, the information provided in the following chapters will help you in your efforts to optimize database calculations:

- Chapter 13, “Designing Partitioned Applications”
- Chapter 25, “Dynamically Calculating Data Values”
- Chapter 27, “Developing Calculation Scripts”
- Chapter 55, “Optimizing with Intelligent Calculation”

Designing for Calculation Performance

You can configure a database to optimize calculation performance.

The best configuration for the site depends on the nature and size of the database. Use the information in the following topics as guidelines only:

- “Block Size and Block Density” on page 1172
- “Order of Sparse Dimensions” on page 1173
- “Incremental Data Loading” on page 1174
- “Database Outlines with Two or More Flat Dimensions” on page 1174
- “Formulas and Calculation Scripts” on page 1174

Block Size and Block Density

A data block size of 8Kb to 100Kb provides optimal performance in most cases.

If data blocks are much smaller than 8Kb, the index is usually very large, forcing Analytic Services to write to and retrieve the index from disk. This process slows down calculation.

If data blocks are much larger than 100Kb, Intelligent Calculation does not work effectively. For a comprehensive discussion of how intelligent calculation aids performance, see Chapter 55, “Optimizing with Intelligent Calculation.”

To optimize calculation performance and data storage, you need to balance data block density and data block size. You can create balance by rearranging the dense and sparse dimension configuration of the database. Therefore, keep these suggestions in mind:

- Keep data block size between 8 Kb and 100 Kb with as high a block density as possible.
- Run test calculations of the most promising configurations of a database that contains representative data. Check the results to determine the configuration that produces the best calculation performance.

You can view information about a database, including the potential and actual number of data blocks and the data block size.

➤ To view data block information, use either of the following methods:

Tool	Topic	Location
Administration Services	Checking Data Block Statistics	*Essbase Administration Services Online Help*
ESSCMD	GETDBINFO	*Technical Reference*

Order of Sparse Dimensions

You may improve calculation performance by changing the order of standard (not attribute) sparse dimensions in the database outline. Order standard sparse dimensions by the number of members they contain, placing the dimension that contains the fewest members first. This arrangement provides a number of possible improvements, depending on the site:

- The calculator cache functions more effectively, providing approximately a 10% performance improvement if you have a database outline with a very large dimension (for example, a dimension containing 1000 members).
- Parallel calculation, if enabled, is more likely to be used if the standard sparse dimension with the most members is the last standard sparse dimension in the outline.

Incremental Data Loading

Many companies load data incrementally. For example, a company may load data each month for that month.

To optimize calculation performance when you load data incrementally, make the dimension tagged as time a sparse dimension. If the time dimension is sparse, the database contains a data block for each time period. When you load data by time period, Analytic Services accesses fewer data blocks because fewer blocks contain the relevant time period. Thus, if you have Intelligent Calculation enabled, only the data blocks marked as dirty are recalculated.

For example, if you load data for March, only the data blocks for March are updated. The data blocks for January and February do not change. With Intelligent Calculation enabled, Analytic Services recalculates only the data blocks for March and the dependent parents of March.

However, making the time dimension sparse when it is naturally dense may significantly increase the size of the index, creating possibly slower performance due to more physical I/O activity to accommodate the large index.

If the dimension tagged as time is dense, you still receive some benefit from Intelligent Calculation when you do a partial data load for a sparse dimension. For example, if Product is sparse and you load data for one product, Analytic Services recalculates only the blocks affected by the partial load, even though time is dense and Intelligent Calculation is enabled.

Database Outlines with Two or More Flat Dimensions

Calculation performance may be affected if a database outline has two or more flat dimensions. A flat dimension has very few parents and each parent has many thousands of children; in other words, flat dimensions have many members and few levels.

You can improve performance for outlines with two or more flat dimensions by adding intermediate levels to the database outline.

Formulas and Calculation Scripts

You may achieve significant improvements in calculation performance by carefully grouping formulas and dimensions in a calculation script. In this way, you can ensure that Analytic Services cycles through the data blocks in the database as few times as possible during a calculation.

Order commands in calculation scripts to make the database calculation as simple as possible. Consider applying all formulas to the database outline and using a default calculation (CALC ALL). This method may improve calculation performance.

For detailed information about developing calculation scripts, see Chapter 27, "Developing Calculation Scripts." For detailed information about multiple calculation passes, see "Calculation Passes" on page 536.

Monitoring and Tracing Calculations

You can display information in the application log about how Analytic Services is calculating the database by using the following commands in a calculation script:

SET MSG SUMMARY and SET MSG DETAIL

You can use the SET MSG SUMMARY and SET MSG DETAIL calculation commands in a calculation script to do the following:

- Display calculation settings, for example, whether completion notice messages are enabled
- Provide statistics on the number of data blocks created, read, and written
- Provide statistics on the number of data cells calculated

The SET MSG DETAIL command also provides a detailed information message every time Analytic Services calculates a data block. SET MSG DETAIL is useful for reviewing the calculation order of data blocks and for testing intelligent recalculations.

CAUTION: Because the SET MSG DETAIL command causes a high processing overhead, use it only during test calculations.

SET MSG SUMMARY causes a processing overhead of approximately 1% to 5%, depending on database size, and is therefore appropriate for all calculations.

SET NOTICE

You can use the SET NOTICE calculation command in a calculation script to display calculation completion notices that tell you what percentage of the database has been calculated. You can use the SET MSG SUMMARY command with the SET NOTICE command to show calculation progress between completion notices. Completion notices do not significantly reduce calculation performance, except when used with a very small database.

Using Simulated Calculations to Estimate Calculation Time

You can simulate a calculation using SET MSG ONLY in a calculation script. A simulated calculation produces results that help you analyze the performance of a real calculation that is based on the same data and outline.

By running a simulated calculation with a command like SET NOTICE HIGH, you can mark the relative amount of time each sparse dimension takes to complete. Then, by performing a real calculation on one or more dimensions, you can estimate how long the full calculation will take, because the time a simulated calculation takes to run is proportional to the time that the actual calculation takes to run.

For example, if the calculation starts at 9:50:00 AM and the first notice is time-stamped at 09:50:10 AM, and the second is time-stamped at 09:50:20 AM, you know that each of part of the calculation took ten seconds. If you then run a real calculation on only the first portion and note that it took 30 seconds to run, you know that the other portion will also take 30 seconds. If there were only two messages total, then you would know that the real calculation will take approximately 60 seconds (20 /10 * 30 = 60 seconds).

In this manner, you can use a simulated calculation to estimate the length of time it takes a calculation to run.

Use the following topics to learn how to perform a simulated calculation and how to use a simulated calculation to estimate calculation time:

- “Performing a Simulated Calculation” on page 1177
- “Estimating Calculation Time” on page 1178
- “Factors Affecting Estimate Accuracy” on page 1179
- “Changing the Outline Based on Results” on page 1180

Performing a Simulated Calculation

Before you can estimate calculation time, you must perform a simulated calculation on a data model that is based on your actual database.

➤ To perform a simulated calculation, use this procedure:

1. Create a data model that uses all dimensions and all levels of detail about which you want information.
2. Load all data. This procedure calculates only data loaded in the database.
3. Create a calculation script with these entries:

   ```
   SET MSG ONLY;
   SET NOTICE HIGH;
   CALC ALL;
   ```

   If you are using dynamic calculations on dense dimensions, substitute the CALC ALL command with the specific dimensions that you need to calculate, for example CALC DIM EAST.

   Note: If you try to validate the script, Analytic Services reports an error. You can disregard the error.

4. Run the script.
5. Find the first sparse calculation message in the application log and note the time in the message.
6. Note the time for each subsequent message.
7. Calculate the dense dimensions of the model that are not being dynamically calculated:

   ```
   CALC DIM (DENSE_DIM1, DENSE_DIM2, …);
   ```

8. Calculate the sparse dimensions of the model:

   ```
   CALC DIM (SPARSEDIM1, SPARSEDIM2, …);
   ```

9. Project the intervals at which notices will occur, and then verify against sparse calculation results. You can then estimate how long a calculation will take.

Estimating Calculation Time

After you perform a simulated calculation, you record the results and use them to estimate actual calculation time.

➤ To estimate the total time required for a calculation, use the following process:

1. Note the times of all the intervals between application log messages generated by SET NOTICE HIGH. See Table 82 for an example.
2. Use the following calculation to estimate the time for a real calculation:

   Total time required for simulated calculation, divided by the first simulated calculation notice interval, multiplied by the first real calculation time interval.

Table 82: Sample Intervals Between Log Messages

Calculation Notice Number	Simulated Calculation Time Interval	Sparse dimension Calculation Interval
1	7 seconds	45 seconds
2	5 seconds	
3	6 seconds	
4	3 seconds	
5	4 seconds	
6	2 seconds	
7	6 seconds	
8	4 seconds	
9	3 seconds	
10	3 seconds	
Total calculation time	43 seconds	

In this example, 43 / 7 * 45 = 276.4 seconds, so the real calculation should take 276.4 seconds.

Factors Affecting Estimate Accuracy

The simulated calculation should return a time accurate to about 5%, excluding the following issues:

- "Variations Due to a Chain of Influences" on page 1179
- "Variations Due to Outline Structure" on page 1179

When these factors are present, this estimating technique more closely predicts calculation time when Analytic Services reaches about 30 to 40 percent of the simulated calculations (30 to 40 percent of the messages generated by SET NOTICE HIGH).

For more information about the SET NOTICE calculation command and the related CALCNOTICE configuration setting, see the *Technical Reference*.

Variations Due to a Chain of Influences

Using SET MSG ONLY as a calculation-time estimating technique should be validated against later CALCNOTICE intervals. The results of this estimating technique vary because of the following chain of influences:

1. Blocks differ in block density through the real consolidation process, therefore
2. The rate at which Analytic Services writes blocks to the disk differs, therefore
3. The rate at which blocks are processed in the cache differs, therefore
4. Actual results may differ from the predicted calculation time.

Variations Due to Outline Structure

Another factor that can make actual results diverge significantly from predicted is the outline structure. Calculations based on CALCNOTICE intervals assume evenly balanced processing time throughout the outline. Factors that can skew this balance include the following situations:

- The model contains 1 or 2 sparse dimensions that are very large in relation to the other sparse dimensions.
- Larger dimensions have member configurations that result in 2 or more shared rollups.

Changing the Outline Based on Results

Once you have estimated and analyzed a simulated calculation, you can make changes in the outline to improve performance.

From top to bottom in the outline, order sparse dimensions to create the fewest percentage increases in upper blocks:

- Level 0 blocks following full model load 100,000
- Upper level blocks after consolidating only sparse dimension 1: 1,000,000
- Upper level blocks after consolidating only sparse dimension 2: 3,000,000
- Upper level blocks after consolidating only sparse dimension 3: 10,000,000
- Upper level blocks after consolidating only sparse dimension 4: 300,000
- Upper level blocks after consolidating only sparse dimension 5: 5,700,000

For example:

- #4 (members = 10,000, 4 levels)
- #1 (members = 500, 2 levels)
- #2 (members = 100, 4 levels)
- #5 (members = 10,000, 4 levels)
- #3 (members = 20, flat)

Use the simulated calculation to generate the upper block count. These numbers may be accurate despite actual dimension sizes as noted next to the items above.

CAUTION: The largest count of members is not always a good predictor.

Estimating Calculation Affects on Database Size

Given the current number of blocks in a database, you can estimate the number of blocks that will be produced by a CALC ALL.

➤ To estimate the database size resulting from a calculation, use the following process (example uses interactive mode):

1. Load data and issue a CALC ALL command and note the average block size.
2. Start the MaxL shell, log into Analytic Services, and start an application and database.

   ```
   essmsh
   login username password;
   alter system load application appname;
   alter application appname load database dbname;
   ```

3. Providing the application and database name, enter the following MaxL statement and note the value that is returned for the number of blocks.

   ```
   query database application.dbname get estimated size;
   ```

4. Multiply the number of blocks by the average size of the blocks in the database.

   Results are accurate to a precision of plus or minus 10%.

Be aware of the following conditions when you query Analytic Services for an estimate of the full size of a database:

- You must perform this query after a CALC ALL. Any other calculation will not produce accurate results.
- You can obtain accurate results with formulas only if they are on sparse dimensions. Estimates on databases that have formulas on dense dimensions are not accurate.
- You cannot obtain accurate results with top down calculations on any member in combination with a lock on data (committed access).
- If you need to estimate partitions, you must query Analytic Services for a database size estimate on every partition and add the results. If you query for the size of only the source database, the estimate includes only the data on the source database server.

Using Parallel Calculation

This topic discusses parallel calculation and enables you to decide whether parallel calculation improves performance for your site. This topic also outlines the process for enabling parallel calculation:

Parallel Calculation

Analytic Services provides two ways of invoking a calculation:

- The calculation may be implicitly specified by the outline itself.
- The calculation may be explicitly specified by a calculation script that you create. The script contains formulas and calculation instructions.

Regardless of how a calculation is triggered, Analytic Services can execute the calculation in one of two modes:

- *Serial calculation* is the default. With serial calculation, each calculation pass is scheduled to run on a single processor. If invoked from a calculation script, the calculations are executed sequentially in the order that they appear in the calculation script.
- *Parallel calculation* breaks each calculation pass into sub-tasks. The sub-tasks that can run independently of each other are scheduled to run simultaneously on up to four threads. Each thread may be on a different processor.

➤ To change from the default serial calculation to parallel calculation, use either of the following methods:

- Change at most two configuration settings and restart the server.
- Add an instruction to the calculation script.

See "Enabling Parallel Calculation" on page 1189 for detailed instructions.

The following topics discuss the details of parallel calculation:

Analytic Services Analysis of Feasibility

Analytic Services evaluates whether using parallel calculation is possible before each calculation pass for which you have enabled parallel calculation.

Analytic Services analyzes the outline and the calculation requested for each calculation pass. Remember that a single calculation may require more than one pass. A number of situations may create the need for more than one pass, including dynamic calculation, the presence of a member tagged as two-pass, or calculations that create certain kinds of inter-dependencies. For a comprehensive discussion of calculation passes, see "Calculation Passes" on page 536.

If Analytic Services determines that parallel calculation is possible, Analytic Services splits the calculation into smaller tasks that are independent of each other. During the calculation, Analytic Services performs the smaller tasks simultaneously.

However, Analytic Services uses serial calculation even if parallel calculation is enabled if there are complex interdependencies between formulas that participate in the pass. Such interdependencies render parallel calculation impossible.

Parallel Calculation Guidelines

Outline structure and application design determine whether enabling parallel calculation can improve calculation performance. Before you enable parallel calculation, review the following guidelines. If you do not adhere to the guidelines, you may not receive the full benefit of parallel calculation:

- Use the uncommitted access isolation level. Parallel calculation is not supported if you use the committed access isolation level. For more information, see "Uncommitted Access" on page 1060.

- One or more formulas present in a calculation may prevent Essbase from using parallel calculation even if it is enabled. For a description of formulas that may force serial calculation regardless of parallel calculation settings, see "Formula Limitations" on page 1185.
- Calculation tasks are usually generated along the last sparse dimension of an outline. Order the sparse dimensions in an outline from smallest to largest, based on actual size of the dimension as reported by the ESSCMD command GETDBSTATS. This ordering recommendation is consistent with recommendations for optimizing calculator cache size and consistent with other outline recommendations. For a description of situations that may need to use additional dimensions (more than the last sparse dimension) and for instructions on how to increase the number of sparse dimensions used, see "Identifying Additional Tasks for Parallel Calculation" on page 1191.
- Parallel calculation is effective on non-partitioned applications and these partitioned applications:
  - Replicated partitions
  - Linked partitions
  - Transparent partitions if the calculation occurs at the target database. The number of sparse dimensions specified by CALCTASKDIMS in the `essbase.cfg` file or by SET CALCTASKDIMS in a calculation script must be set at 1 (the default value). See "Transparent Partition Limitations" on page 1186 for details about the limitations imposed by the use of parallel calculation with transparent partitions, and see "Identifying Additional Tasks for Parallel Calculation" on page 1191 for details about using CALCTASKDIMS or SET CALCTASKDIMS.
- If you have selected incremental restructuring for a database and you have made outline changes that are pending a restructure, do not use parallel calculation. Unpredictable results may occur.

Relationship Between Parallel Calculation and Other Analytic Services Features

The following topics discuss the relationship between parallel calculation and other Analytic Services functionality:

- "Retrieval Performance" on page 1185
- "Formula Limitations" on page 1185

Retrieval Performance

Placing the largest sparse dimension at the end of the outline for maximum parallel calculation performance may slow retrieval performance. See "Optimizing Query Performance" on page 100 for outline design guidelines for optimizing retrieval.

Formula Limitations

The presence of some formulas may force serial calculation. The following formula placements are likely to force serial calculation:

- A formula on a dense member, including all stored members and any Dynamic Calc members upon which a stored member may be dependent, that causes a dependence on a member of the dimension that is used to identify tasks for parallel calculation.
- A formula that contains references to variables declared in a calculation script that uses @VAR, @ARRAY, or @XREF.
- A member formula that causes a circular dependence. For example, member A has a formula referring to member B, and member B has a formula referring to member C, and member C has a formula referring to member A.
- A formula on a dense or sparse member with a dependency on a member or members from the dimension used to identify tasks for parallel processing.

If you need to use a formula that might prevent parallel calculation, you can either mark the member of the formula as Dynamic Calc or exclude it from the scope of the calculation. To check if a formula is preventing parallel calculation, check the application log. See "Monitoring Parallel Calculation" on page 1192 for details about the relevant error messages.

Calculator Cache

At the start of a calculation pass, Analytic Services checks the calculator cache size and the degree of parallelism and then uses the calculator cache bitmap option appropriate for maximum performance. Therefore, the bitmap option used for parallel calculation may be different from the one used for serial calculation.

For example, assume Essbase performs a serial calculation and uses multiple bitmaps and a single anchoring dimension. Without explicit change of the calculator cache size, Analytic Services might perform a parallel calculation might using only a single bitmap and a single anchoring dimension.

You can determine the calculator cache mode that controls the bitmap options by checking the application log at the start of each calculation pass for an entry similar to the following:

```
Multiple bitmap mode calculator cache memory usage has a limit
of [50000] bitmaps.
```

When using parallel calculation in multiple bitmap mode, you may encounter high memory usage. If you encounter this situation, you can use the configuration setting PARCALCMULTIPLEBITMAPMEMOPT to optimize memory use in multiple bitmap mode. This setting can be used together with, or separately from, MULTIPLEBITMAPMEMCHECK. To enable PARCALCMULTIPLEBITMAPMEMOPT, add this line to your `essbase.cfg` file:

```
PARCALCMULTIPLEBITMAPMEMOPT TRUE
```

For a comprehensive discussion of calculator cache and calculator cache bitmaps, see “Sizing the Calculator Cache” on page 1133.

Transparent Partition Limitations

Parallel calculation with transparent partitions has the following limitations:

- You cannot use parallel calculation across transparent partitions unless the calculation occurs at the target.
- You must set CALCTASKDIMS or SET CALCTASKDIMS to 1 (the default) so that there is only one anchoring dimension.

- You must increase the calculator cache so that multiple bitmaps can be used. You can identify the calculator cache mode that controls the bitmap options by checking the application log at the start of each calculation pass for an entry similar to the following:

  ```
  Multiple bitmap mode calculator cache memory usage has a
  limit of [50000] bitmaps.
  ```

  For a comprehensive discussion of sizing the calculator cache, see "Sizing the Calculator Cache" on page 1133.

Restructuring Limitation

Do not use parallel calculation if you have selected incremental restructuring. Parallel calculation does not support incremental restructuring.

Commit Threshold Adjustments

Essbase checks the commit threshold specified by the database setting "Number of blocks before internal commit." If the setting requires less than 10 MB of data be written before an internal commit, then Essbase automatically increases the commit threshold for the duration of the calculation pass to 10 MB. If the setting is greater than 10 MB, Analytic Services uses the setting value.

Analytic Services writes a message to the application log noting the temporary increase if it occurs.

If you can allocate more than 10 MB extra disk space for calculation, consider increasing the commit threshold value, that is, the number of blocks before a commit, to a very large number for better performance.

➤ To view the current threshold, use any of the following methods:

Tool	Topic	Location
Administration Services	Setting Data Integrity Options	*Essbase Administration Services Online Help*
MaxL	**display database** *dbs_name*	*Technical Reference*
ESSCMD	GETDBINFO: Number of blocks modified before internal commit	*Technical Reference*

➤ To modify the commit threshold, use any of the following methods:

Tool	Topic	Location
Administration Services	Setting Data Integrity Options	*Essbase Administration Services Online Help*
MaxL	**alter database *dbs_name* set implicit_commit after *n* blocks**	*Technical Reference*, list of MaxL statements
ESSCMD	SETDBSTATEITEM 21	"Example of Specifying Isolation Level Settings with ESSCMD" on page 1065

For a discussion of commit thresholds, see "Uncommitted Access" on page 1060.

Isolation Level Limitation

You must use uncommitted mode for parallel calculation.

➤ To set the isolation level to uncommitted mode, use any of the following methods:

Tool	Topic	Location
Administration Services	Setting Data Integrity Options	*Essbase Administration Services Online Help*
MaxL	**alter database *dbs_name* disable committed_mode**	*Technical Reference*, list of MaxL statements
ESSCMD	SETDBSTATEITEM 18	"Example of Specifying Isolation Level Settings with ESSCMD" on page 1065

See "Uncommitted Access" on page 1060 for a description of how uncommitted mode works.

Checking Current Parallel Calculation Settings

You can check either the server configuration file or the calculation script that you plan to use to see if parallel calculation is enabled.

➤ To check if parallel calculation has already been enabled in the server configuration file:

1. Open the server `essbase.cfg` file with a text editor.
2. Search for the parameter CALCPARALLEL, and check its specified value.

   The number of threads that can simultaneously perform tasks to complete a calculation is specified by the value 1–4. See the *Technical Reference* for details.

➤ To check if a calculation script sets parallel calculation, look for the SET CALCPARALLEL command. Review the script carefully, as the script may enable or disable parallel calculation more than once.

Enabling Parallel Calculation

To use parallel calculation, enable it at the server level, application level, or database level using either of these methods:

- Add or edit the appropriate configuration settings to the `essbase.cfg` file.

  For instructions, see CALCPARALLEL and CALCTASKDIMS in the configuration settings section of the *Technical Reference*.

- Add the appropriate calculation commands to a calculation script.

  For instructions, see SET CALCPARALLEL and SET CALCTASKDIMS in the calculator commands section of the *Technical Reference*.

Parallel calculation settings use standard precedence rules:

- The database setting takes precedence over the application setting
- The application setting takes precedence over the server setting.

Setting parallel calculation at the server level enables it for all calculations performed on all applications and databases on the server. You can disable parallel calculation for individual applications or databases by setting parallel calculation at the server level in the configuration file and then adding application-specific or database-specific entries in a calculation script.

CAUTION: Be sure to read all of this chapter before attempting to enable parallel calculation.

➤ To enable parallel calculation, use the following process:

1. If you plan to enable parallel calculation in the configuration file, check the current status to see if an entry already exists; use the process described in "Checking Current Parallel Calculation Settings" on page 1189.
2. Add or modify CALCPARALLEL to the `essbase.cfg` file on the server, or add SET CALCPARALLEL to a calculation script.
3. If needed, enable Analytic Services to use more than the one sparse dimension to identify tasks for parallel calculation; use the process described in "Identifying Additional Tasks for Parallel Calculation" on page 1191.
4. If you added entries to the configuration file, restart the server.
5. Run the calculation.

Hyperion recommends that you set the value of CALCPARALLEL to be one less than the number of processors available for calculation. This extra processor can then be used by either the operating system or by the Essbase process responsible for writing out dirty blocks from the cache.

Tip: You can combine the use of CALCPARALLEL and SET CALCPARALLEL if the site requires it. For example, you can set CALCPARALLEL as off at the server level, and use a calculation script to enable and disable parallel calculation as often as needed.

Identifying Additional Tasks for Parallel Calculation

By default, Analytic Services uses the last sparse dimension in an outline to identify tasks that can be performed concurrently. But the distribution of data may cause one or more tasks to be empty, that is, there are no blocks to be calculated in the part of the database identified by a task. This situation can lead to uneven load balancing, thus reducing the effectiveness of parallel calculation.

To resolve this situation, you can enable Analytic Services to use additional sparse dimensions in the identification of tasks for parallel calculation. For example, if you have a FIX statement on a member of the last sparse dimension, you can include the next-to-last sparse dimension from the outline as well. Because each unique member combination of these two dimensions is identified as a potential task, more and smaller tasks are created, increasing the opportunities for parallel processing and providing better load balancing.

➤ To increase the number of sparse dimensions used to identify tasks for parallel calculation, use the following process:

1. If you are not sure, verify if parallel calculation is already enabled. See "Checking Current Parallel Calculation Settings" on page 1189 for instructions. Without CALCPARALLEL (or SET CALCPARALLEL in a calculation script), CALTASKDIMS has no effect.
2. Add or modify CALCTASKDIMS in the `essbase.cfg` file on the server, or use the calculation script command SET CALCTASKDIMS at the top of the script. See the *Technical Reference* for instructions.
3. If you add or modify CALCTASKDIMS in the `essbase.cfg` file on the server, restart Analytic Services.
4. If you are using a calculation script, run the script.

Note: In some cases, Analytic Services uses a lower number of dimensions to identify tasks than is specified by CALCTASKDIMS or SET CALCTASKDIMS. See the *Technical Reference* for a detailed explanation.

Monitoring Parallel Calculation

You can view events related to parallel calculation in the application log:

➤ To view the application log, see "Viewing Logs" in *Essbase Administration Services Online Help*.

For each calculation pass, Analytic Services writes several types of information to the application log to support parallel calculation:

- If you have enabled parallel calculation and Analytic Services has determined that parallel calculation can be performed, Analytic Services writes a message in the application log:

  ```
  Calculating in parallel with n threads
  ```

  In this message, *n* represents the number of concurrent tasks specified in CALCPARALLEL or SETCALCPARALLEL.

- For each formula that prevents parallel calculation (forces serial calculation), Analytic Services writes a message to the application log:

  ```
  Formula on mbr memberName prevents calculation from
  running in parallel.
  ```

  In the message, *memberName* represents the name of the member where the relevant formula exists. You can look in the application log for such messages and consider tagging the relevant member or members as Dynamic Calc if possible so they do not feature in the calculation pass.

- Analytic Services writes a message to the application log specifying the number of tasks that can be executed concurrently at any one time (based on the data, not the value of CALCPARALLEL or SETCALCPARALLEL):

  ```
  Calculation task schedule [576,35,14,3,2,1]
  ```

  The example message indicates that 576 tasks can be executed concurrently. After the 576 tasks complete, 35 more can be performed concurrently, and so on.

  The benefit of parallel calculation is greatest in the first few steps, and then taper off because there are fewer and fewer tasks being performed concurrently.

  The degree of parallelism depends on the number of tasks shown in the task schedule. The greater the number, the more tasks that can run in parallel, and thus the greater the performance gains.

- Analytic Services writes a message to the application log indicating how many of the tasks are empty (contain no calculations):

  ```
  [Tue Nov 27 12:30:44 2001]Local/CCDemo/Finance/essexer/
  Info(1012681) Empty tasks [91,1,0,0]
  ```

  In the example log message, Analytic Services indicates that 91 of the tasks at level zero were empty.

  If the ratio of empty tasks to the tasks specified in the task schedule is greater than 50, parallelism may not be giving you improved performance, because of the high sparsity in the data model.

  You can change dense/sparse assignments to reduce the number of empty tasks, and increase the performance gains from parallel calculation.

Using Formulas

You may achieve significant improvements in calculation performance by careful use of formulas in the database outline. For example, you may achieve improved calculation performance by placing formulas on members in the database outline instead of placing the formulas in a calculation script. For a comprehensive discussion of how to develop formulas, see Chapter 22, “Developing Formulas.”

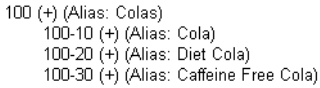
➤ For discussion of how to handle formula issues that affect performance, see the following topics:

- “Consolidating” on page 1194
- “Using Simple Formulas” on page 1194
- “Using Complex Formulas” on page 1195
- “Optimizing Formulas on Sparse Dimensions in Large Database Outlines” on page 1196
- “Constant Values Assigned to Members in a Sparse Dimension” on page 1196
- “Using Cross-Dimensional Operators (->)” on page 1198
- “Bottom-Up and Top-Down Calculation” on page 1200

Consolidating

Using the database outline to roll up values is always more efficient than using a formula to calculate values. For example, consider the consolidation on the Sample Basic database outline shown in Figure 244.

Figure 244: Consolidation on Sample Basic Outline



Using outline consolidation is more efficient than applying the following formula to the 100 member:

```
100-10 + 100-20 + 100-30
```

Using Simple Formulas

If you use a simple formula and block size is not unusually large, you can place the formula on a member of either a sparse or a dense dimension without significantly affecting calculation performance. The bigger the block size, the more impact simple formulas have on calculation performance. For a discussion of the relationship between block size and calculation performance, see "Block Size and Block Density" on page 1172.

A simple formula is, for example, a ratio or a percentage. A simple formula meets all of the following requirements:

- Does not reference values from a different dimension (sparse or dense). For example, a simple formula cannot reference Product->Jan.
- Does not use range functions. For example, a simple formula cannot use @AVGRANGE, @MAXRANGE, @MINRANGE, or @SUMRANGE.
- Does not use relationship or financial functions. For example, a simple formula cannot use @ANCESTVAL, @NEXT, @PARENTVAL, @SHIFT, @ACCUM, or @GROWTH. For a complete list of relationship and financial functions, see the *Technical Reference*.

For information on how formulas affect calculation performance, see "Bottom-Up and Top-Down Calculation" on page 1200.

Using Complex Formulas

If you use a complex formula, you can improve performance by applying the following guidelines:

- If possible, apply the formula to a member in a *dense* dimension.
- Use the FIX command in a calculation script to calculate only required data blocks. For an example of the use of FIX, see "Using the FIX Command" on page 598.
- Increase the density of the database (ratio of existing data blocks to possible data blocks). For a discussion of balance between block size and block density, see "Block Size and Block Density" on page 1172.

A complex formula is a formula that meets any of the following requirements:

- References a member or members in a different dimension (sparse or dense); for example, Product->Jan.
- Uses one or more range functions, for example, @AVGRANGE, @MAXRANGE, @MINRANGE, or @SUMRANGE.
- Uses relationship or financial functions; for example, @ANCESTVAL, @NEXT, @PARENTVAL, @SHIFT, @ACCUM, or @GROWTH. For a complete list of relationship and financial functions, see the *Technical Reference*.

When applied to sparse dimension members, complex formulas create more calculation overhead and therefore slow performance. This problem occurs because the presence of complex formulas requires Essbase to perform calculations on all possible as well as all existing data blocks related to the member with the complex formula. The presence of a relationship or financial function on a sparse dimension member causes Essbase to perform calculations on all blocks, possible as well as existing, increasing the overhead even more.

Thus, a complex formula that includes a relationship or financial function creates a greater overhead increase than does a complex formula that does not include a relationship or financial function.

For a comprehensive discussion about how complex formulas affect calculation performance, see "Bottom-Up and Top-Down Calculation" on page 1200.

Two examples illustrate complex formula overhead:

- If a database has 90 existing data blocks and 100 potential data blocks, the overhead for complex formulas is not very large, not more than ten extra blocks to read and possibly write values to.
- If a database has 10 existing data blocks and 100 potential data blocks, the overhead is as much as ten times what it would be without the complex formula (depending on the particular outline structure and other factors), as many as 90 extra blocks to read and possibly write to.

In all cases, the lower the ratio of existing data blocks to possible data blocks, the higher the calculation performance overhead and the slower the performance.

Optimizing Formulas on Sparse Dimensions in Large Database Outlines

You can use the SET FRMLBOTTOMUP calculation command to optimize the calculation of formulas in sparse dimensions in large database outlines. With this command, you can force a bottom-up calculation on sparse member formulas that would otherwise be calculated top-down. For a review of methods and a caution, see "Forcing a Bottom-Up Calculation" on page 1202.

Forcing a bottom-up calculation on a top-down formula enables efficient use of the CALC ALL and CALC DIM commands. For technical details, review the discussions of the SET FRMLBOTTOMUP calculation command and the CALCOPTFRMLBOTTOMUP configuration setting in the *Technical Reference*.

Constant Values Assigned to Members in a Sparse Dimension

If you assign a constant to a member in a sparse dimension, Analytic Services automatically creates a data block for every combination of sparse dimension members that contains the member.

For example, assume that a member or a calculation script formula contains the following expression:

```
California = 120;
```

In this formula, California is a member in a sparse dimension and 120 is a constant value. Analytic Services automatically creates all possible data blocks for California and assigns the value 120 to all data cells. Many thousands of data blocks may be created. To improve performance, create a formula that does not create unnecessary values.

➤ To assign constants in a sparse dimension to only those intersections that require a value, use FIX in a manner similar to the following example:

```
FIX(Colas,Misc,Actual)
California = 120;
ENDFIX
```

In this example, Colas is a member of the sparse dimension, Product; Actual is a member of the dense dimension, Scenario; and Misc is a member of the dense dimension, Measures. The value 120 is assigned to any intersection of California (in the Market dimension), Actual (in the Scenario dimension), Misc (in the Measures dimension), Colas (in the Product dimension), and any member in the Year dimension, because a specific member of Year is not specified in the script.

Because Sample Basic includes only two sparse dimensions, this example affects only one block. If there were additional sparse dimensions, Essbase would ensure that there are data blocks for all combinations of the sparse dimensions with California and Colas, creating new blocks if they do not exist. Within the new blocks, Analytic Services sets Measures and Scenario values (other than those assigned the value 120) to #MISSING.

Non-Constant Values Assigned to Members in a Sparse Dimension

If you assign non-constant values to members of a sparse dimension, new blocks are created based on the Create Blocks on Equations setting. The Create Blocks on Equations setting is defined at the database level, as a database property. Within calculation scripts you can temporarily override the Create Blocks on Equations setting. (For basic information about this setting, see “Non-Constant Values” on page 478.) Consider the effects of the following calculation when West has no value and the Create Blocks on Equations setting is ON.

```
West = California + 120;
```

Unneeded blocks may be created for all sparse-member intersections with West, even if the corresponding block value is #MISSING for all of the children of West. Especially in a large database, creation and processing of unneeded blocks requires additional processing time.

➤ To control creation of blocks when you assign non-constant values to members of a sparse dimension, use the SET CREATEBLOCKONEQ ON|OFF command. The following script includes calculations with this setting off and on.

```
FIX (Colas);
SET CREATEBLOCKONEQ OFF
West = California + 120;
SET CREATEBLOCKONEQ ON
East = “New York” + 100;
ENDFIX
```

Because the Create Block on Equation setting is disabled at the beginning, West blocks are created only when values exist for the children of West. Later, because the Create Block on Equation setting is enabled, all blocks for East are created.

Note: Use of SET CREATEBLOCKONEQ affects only creation of blocks during the execution of the calculation script that contains this command. This command does not change the overall database Create Blocks on Equations setting.

For details regarding use of SET CREATEBLOCKEQ ON|OFF in calculation scripts, see the *Technical Reference*.

Using Cross-Dimensional Operators (->)

Use caution when using a cross-dimensional operator (->) in the following situations:

- “On the Left Side of an Equation” on page 1198
- “In Equations in a Dense Dimension” on page 1199

On the Left Side of an Equation

For faster calculation script performance, use FIX in the calculation script to qualify the use of a formula instead of using a formula that includes a cross-dimensional operator on the left side of an equation.

For example, assume that you want to increase the Jan -> Sales values in Sample Basic by 5%. To improve performance by calculating only the relevant combinations of members, use the FIX command in a calculation script:

```
FIX(Jan)
   Sales = Sales * .05;
ENDFIX
```

With the FIX command, Analytic Services calculates the formula only for specified member combinations, in this example, for combinations that include Jan.

Compare this technique to using the slower cross-dimensional operator approach. For the previous example, you place the following formula on the Sales member in the database outline:

```
Sales(Sales -> Jan = Sales -> Jan * .05;)
```

As Analytic Services cycles through the database, it calculates the formula for every member combination that includes a member from the dimension tagged as time (Jan, Feb, Mar, etc.), even though only January combinations need to be calculated.

For detailed information on calculation scripts and the FIX command, see "Using the FIX Command" on page 598 and the *Technical Reference*.

In Equations in a Dense Dimension

When you use a cross-dimensional operator in an equation in a dense dimension, Analytic Services does not automatically create the required blocks if both of these conditions apply:

- Resultant values are from a dense dimension.
- The operand or operands are from a sparse dimension.

You can use the following techniques to create the blocks and avoid the performance issue.

- Ensure that the results members are from a sparse dimension, not from a dense dimension. In this example, the results member Budget is from a sparse dimension:

  ```
  FIX(Sales)
        Budget = Actual * 1.1;
  ENDFIX
  ```

```
FIX(Expenses)
      Budget = Actual *  .95;
ENDFIX
```

- Use the DATACOPY calculation command to create and then calculate the required blocks. See “Using DATACOPY to Copy Existing Blocks” on page 600 for more and example and more information about using this command for this purpose.
- Use a member formula that contains the dense member equations:

```
FIX(Sales, Expenses)
Budget (Sales = Sales -> Actual * 1.1;
Expenses = Expenses -> Actual * .95;)
ENDFIX
```

Using Bottom-Up Calculation

A top-down calculation is less efficient than a bottom-up calculation because more blocks are calculated than is necessary. Although a top-down calculation is less efficient than a bottom-up calculation, top-down calculations are necessary in some cases to ensure that calculation results are correct.

Use the following topics to determine whether bottom-up and top-down calculation is more appropriate for a particular situation:

- “Bottom-Up and Top-Down Calculation” on page 1200
- “Forcing a Bottom-Up Calculation” on page 1202

Bottom-Up and Top-Down Calculation

Analytic Services uses one of two calculation methods to do a full calculation of a database outline—bottom-up calculation or top-down calculation. By default, Analytic Services does a bottom-up calculation of a database.

For a bottom-up calculation, Analytic Services determines which data blocks need to be calculated before it calculates the database. Analytic Services then calculates only the blocks that need to be calculated. The calculation begins with the existing block with the lowest block number and works up through each block in number order until the existing block with the highest block number is reached. For a detailed explanation of block calculation order, see “Block Calculation Order” on page 524.

If the database outline contains a complex member formula, Analytic Services performs a top-down calculation for the relevant member.

Use the following information to learn more about simple and complex formula interactions with bottom-up and top-down calculation:

- “Bottom-Up Calculations and Simple Formulas” on page 1201
- “Top-Down Calculations and Complex Formulas” on page 1201

Bottom-Up Calculations and Simple Formulas

For simple formulas, Analytic Services does a bottom-up calculation to determine which blocks need to be calculated prior to running the full calculation. For example, for a simple formula on a member (such as A = B + C), A is calculated only if B or C exists in the database. That is, the dependency of the formula on B and C is known before the calculation is started.

Top-Down Calculations and Complex Formulas

Before starting a calculation, Analytic Services searches the database outline and marks complex formulas that require top-down calculation; for example, a member formula that contains a cross-dimensional reference. When Analytic Services reaches a member with a top-down formula, it does a top-down calculation for the member.

When a formula on a member is complex, all possible blocks for the member must be examined to see if an existing block needs to be changed or a new block needs to be created; it is difficult to determine the dependency that blocks may have on other blocks prior to the start of the calculation. The top-down method slows calculation performance because Analytic Services must search for appropriate blocks to calculate in order to execute the formula.

When a formula is compiled, if the formula is to be calculated top-down, Analytic Services logs a message in the application log file.

Consider the following complex formula:

```
A = B -> D + C -> D
```

To calculate the formula, Analytic Services must examine every possible combination of A to see whether B -> D or C -> D exists.

For descriptions and examples of complex formulas, see “Using Complex Formulas” on page 1195.

Forcing a Bottom-Up Calculation

If it is appropriate for the site, you can force a bottom-up calculation on a top-down formula.

➤ To force a bottom-up calculation, use any of the following methods:

Method	Topic Where Discussed	Location
Calculation function	@CALCMODE in a formula	*Technical Reference*
Calculation script command	SET FRMLBOTTOMUP	*Technical Reference*
`essbase.cfg` file setting	CALCOPTFRMLBOTTOMUP or CALCMODE	*Technical Reference*

Forcing a bottom-up calculation on a formula ordinarily increases performance time. If the formula contains complex functions (for example, range functions) or if the formula's dependencies are not straightforward, a bottom-up calculation may produce results that differ from the results of a top-down calculation.

CAUTION: Before changing the setting CALCOPTFRMLBOTTOMUP or using the calculation script command SET FRMLBOTTOMUP in a production environment, check the validity of calculation results by comparing, relative to the same data, the results of a bottom-up calculation and the results of a top-down calculation.

Managing Caches to Improve Performance

The following section describes the caches that are used with block storage databases. For information about the aggregate storage cache, see "Managing the Aggregate Storage Cache" on page 1345.

When calculating a database, Analytic Services uses approximately 30 bytes of memory per member in the database outline. So if the database has 5,000 members, Analytic Services needs approximately 150K of memory to calculate the database.

Note: When you run concurrent calculations, each calculation uses separate memory space. For example, if you are running two calculation scripts concurrently, Analytic Services requires 60 bytes per member: 30 bytes per member per script. Concurrent calculations use the same memory caches. You can avoid excess memory use by combining calculation scripts. You can obtain good performance by using parallel calculation with a single calculation script. For a comprehensive discussion of parallel calculation, see "Using Parallel Calculation" on page 1182.

Analytic Services uses memory to optimize calculation performance, especially for large calculations. The amount of memory used is not controllable, except by altering the size of the database outline. However, you can ensure that the memory cache sizes enable Analytic Services to optimize the calculation.

Analytic Services uses memory caches to coordinate memory usage:

- The calculator cache. Ensure that the calculator cache is large enough to optimize calculation performance.
- The dynamic calculator cache
- The index cache. If the database is large, the default index cache is not large enough to provide optimum calculation performance.
- The data cache.
- The data file cache.

Note: When you calculate a database for the first time, the size of the calculator cache is particularly significant for calculation performance. If possible, ensure that the calculator cache is large enough for Essbase to use the optimal calculator cache option.

For a comprehensive discussion of cache sizing, see "Sizing Caches" on page 1128. Make sure that you read the entire topic before making any changes.

Working with the Block Locking System

When a block is calculated, Analytic Services locks the block and all blocks that contain the children of the block. Analytic Services calculates the block and then releases both the block and the blocks containing the children.

By default, Analytic Services locks up to 100 blocks concurrently when calculating a block. This number of block locks is sufficient for most database calculations. If you are calculating a formula in a sparse dimension, Analytic Services works most efficiently if it can lock all required child blocks concurrently. Therefore, when calculating a formula in a sparse dimension, you may want to set a lock number higher than 100 if you are consolidating very large numbers of children (for example, more than 100 children). By increasing the number, you ensure that Analytic Services can lock all required blocks, and therefore, performance is not impaired.

Analytic Services locking behavior depends on the isolation level setting. For a detailed explanation of locking behavior, see "Locking Under Committed Access" on page 1057 and "Locking Under Uncommitted Access" on page 1061.

Note: For consolidations in a sparse dimension, block locking is not a consideration because Analytic Services does not need to lock all blocks containing children concurrently.

Using SET LOCKBLOCK and CALCLOCKBLOCK

You can use the SET LOCKBLOCK command in a calculation script along with the CALCLOCKBLOCK setting in the `essbase.cfg` file to specify the maximum number of blocks that Analytic Services can lock concurrently when calculating a block. If you do not modify the default setting and the default 100 blocks is not sufficient during calculation, the calculation may be require more time than expected.

Managing Concurrent Access for Users

Analytic Services uses the block locking system to manage concurrent access to users. This system ensures that only one user at a time can update or calculate a particular data block. How Analytic Services handles locking blocks and committing data depends on the isolation level setting.

When Analytic Services calculates a data block, it creates an exclusive lock. Thus, no other user can update or calculate the data block. However, other users can have read-only access to the block. When Analytic Services finishes the calculation, it releases the block. Other users can then update the block if they have the appropriate security access.

When a user is updating a data block, the block is locked. If a database calculation requires a data block that is being updated by another user, the calculation waits for one of the following:

- For the data block to be released if the isolation level setting is Uncommitted Access.
- For the calculation to complete if the isolation level setting is Committed Access.

Analytic Services does not provide a message to say that the calculation is waiting for the data block to be released.

You can prevent calculation delays caused by waiting for locked blocks by using Analytic Services security options to do either of the following:

- Deny access to other users
- Disconnect users from Analytic Services

For detailed information about and instructions for the security options, see "Disconnecting Users and Terminating Requests" on page 856 and "Managing Passwords and User Names" on page 859.

For information on how Analytic Services handles locks and transactions, see "Understanding How Analytic Services Handles Transactions" on page 1064 and "Data Locks" on page 1054.

Note: When Analytic Services locks a block for calculation, it does not put an exclusive lock on the dependent child blocks. Thus, another user can update values in the child blocks. If necessary, you can use the above security options to prevent such updates.

Using Two-Pass Calculation

You can improve performance significantly by tagging an accounts dimension member as two-pass in the database outline, if it is appropriate for the application. The combination of data and calculation needs may require the use of a calculation script to calculate a formula twice, instead of two-pass tagging, to preserve accuracy.

Use these sections to understand more about two-pass calculation, and decide whether you can tag an accounts dimension member as two-pass to improve performance or whether you must use a calculation script to calculate a formula twice. This section also provides information about how to enable two-pass calculation or create a calculation script for two-pass calculation:

For information about the interaction of two-pass calculation and attribute members, see Table 12 on page 188.

Understanding Two-Pass Calculation

You can use a two-pass calculation on member formulas that need to be calculated twice to produce the correct value.

Whenever possible, Analytic Services calculates two-pass formulas at the data block level, calculating the two-pass formulas at the same time as the main calculation. Thus, Analytic Services does not need to do an extra calculation pass through the database. However, in some situations, Analytic Services needs an extra calculation pass through the database.

How Analytic Services calculates the two-pass formulas depends on whether there is a dimension tagged as time as well as a dimension tagged as accounts. It also depends on the dense-sparse configuration of the time and account dimensions.

Reviewing a Two-Pass Calculation Example

For example, consider this calculation required for Profit%:

```
Profit % = Profit % Sales
```

Assume that the following table shows a subset of a data block with Measures and Year as dense dimensions. Measures is tagged as accounts, and Year is tagged as time. The AGGMISSG setting is turned off (the default).

Data values have been loaded into the input cells. Analytic Services calculates the shaded cells. The numbers in bold show the calculation order for the cells. Cells with multiple consolidation paths are darkly shaded.

Measures -> Year	Jan	Feb	Mar	Qtr1
Profit	75	50	120	**5**
Sales	150	200	240	**6**
Profit%	**1**	**2**	**3**	**4/ 7**

Note: For detailed information on how cell calculation order depends on database configuration, see "Cell Calculation Order" on page 527.

Analytic Services uses this calculation order:

1. Analytic Services calculates the formula `Profit % Sales` for Profit % -> Jan, Profit % -> Feb, Profit % -> Mar, and Profit % -> Qtr1 (1, 2, 3, 4 above).
2. Analytic Services calculates Profit -> Qtr1 and Sales -> Qtr1 by adding the values for Jan, Feb, and Mar (5, 6 above).
3. Analytic Services calculates Profit % -> Qtr1 by adding the values for Profit % -> Jan, Profit % -> Feb, and Profit % -> Mar (7 above). This addition of percentages produces the value %125, not the correct result.

Measures/Year	Jan	Feb	Mar	Qtr1
Profit	75	50	120	**245 (5)**
Sales	150	200	240	**590 (6)**
Profit%	**50% (1)**	**25% (2)**	**50% (3)**	**0% (4)** **125% (7)**

4. If you tag Profit % as two-pass in the database outline, Analytic Services uses the `Profit % Sales` formula to recalculate the Profit % values and produce the correct results.

Measures/Year	Jan	Feb	Mar	Qtr1
Profit	75	50	120	**245 (5)**
Sales	150	200	240	**590 (6)**
Profit%	**50% (1)**	**25% (2)**	**50% (3)**	**0% (4)** **125% (7)** **42% (8)**

For detailed information about multiple calculation passes, see "Calculation Passes" on page 536.

Understanding the Interaction of Two-Pass Calculation and Intelligent Calculation

Two scenarios are described in detail in the following sections. If you are using Intelligent Calculation, use the scenario that matches the configuration of the database; each scenario tells you how to ensure that Analytic Services calculates two-pass formulas accurately.

These scenarios require that you understand the concepts of Intelligent Calculation. For a comprehensive discussion, see Chapter 55, "Optimizing with Intelligent Calculation."

Scenario A

Scenario A demonstrates two key approaches:

- "No Extra Calculation Pass for Two-Pass Formulas" on page 1209
- "All Data Blocks Marked As Clean" on page 1209

In this scenario, you place formulas in the outline and then, as appropriate, tag specific formulas as two-pass for best performance.

No Extra Calculation Pass for Two-Pass Formulas

Analytic Services calculates the two-pass formulas while it is calculating the data block. Thus, Analytic Services does not need to do an extra calculation pass through the database.

All Data Blocks Marked As Clean

After the calculation, all data blocks are marked as clean for the purposes of Intelligent Calculation.

When you tag a member formula as two-pass in the outline, Analytic Services does the two-pass calculation while each data block is being calculated. However, when you repeat a formula in a calculation script, Analytic Services has to read the data blocks and write them to memory in order to recalculate the formula.

Scenario B

Scenario B illustrates two key approaches:

- “Extra Calculation Pass for Two-Pass Formulas” on page 1209
- “Data Blocks for Two-pass Formulas Not Marked As Clean” on page 1209

In this scenario, you create a calculation script to perform the formula calculation for best performance.

Extra Calculation Pass for Two-Pass Formulas

Analytic Services calculates the database and then does an extra calculation pass to calculate the two-pass formulas. Even though all data blocks are marked as clean after the first database calculation, Analytic Services ignores the clean status on the blocks that are relevant to the two-pass formula and recalculates these blocks.

Data Blocks for Two-pass Formulas Not Marked As Clean

After the first calculation, Analytic Services has marked all the data blocks as clean for the purposes of Intelligent Calculation. In a second calculation pass through the database, Analytic Services recalculates the required data blocks for the two-pass formulas. However, because the second calculation is a partial

calculation of the database, Analytic Services does not mark the recalculated blocks as clean. When you recalculate the database with Intelligent Calculation turned on, these data blocks may be recalculated unnecessarily.

If the database configuration allows Analytic Services to use Scenario B, consider using a calculation script to perform two-pass formula calculations. If you use a calculation script, Analytic Services still does an extra calculation pass through the database; however, you can ensure that Analytic Services has marked all the data blocks as clean after the calculation. For a review of methods, see "Creating Calculation Scripts for Two-Pass and Intelligent Calculation" on page 1212.

Choosing Two-Pass Calculation Tag or Calculation Script

Even though tagging an accounts member as two-pass may bring performance benefits, some applications cannot use this method. Check these qualifications to see whether you should apply a two-pass tag or create a calculation script that performs a calculation twice for best performance and accuracy:

- You can tag a member as two-pass if it is in a dimension tagged as accounts. When you perform a default calculation on the database, Analytic Services automatically recalculates any formulas tagged as two-pass if they are in the dimension tagged as accounts in the database outline.
- You can tag a member as two-pass if it is a Dynamic Calc or Dynamic Calc and Store member of any dimension. For a comprehensive discussion of dynamic calculation, see Chapter 25, "Dynamically Calculating Data Values."
- You may need to use a calculation script to calculate a two-pass formula to obtain accurate results, even if the two-pass tag would provide performance benefits. For a review of methods, see "Creating Calculation Scripts for Two-Pass and Intelligent Calculation" on page 1212.
- Use a calculation script instead of the two-pass tag to ensure efficient use of Intelligent Calculation. For a detailed explanation of two scenarios, see "Understanding the Interaction of Two-Pass Calculation and Intelligent Calculation" on page 1208.
- You need to use a calculation script to calculate a formula twice if the database configuration means that Analytic Services uses Scenario A, as described in "Scenario A" on page 1208, and if the formula references values from another data block.

- You may want to use a calculation script to calculate two-pass formulas if the database configuration means that Analytic Services uses Scenario B, as described in "Scenario B" on page 1209.

Enabling Two-Pass on Default Calculations

A database setting enables two-pass calculation in default calculations. When you perform a default calculation on a database with two-pass calculation enabled (the default setting), Analytic Services automatically attempts to calculate any formulas tagged as two-pass in the dimension tagged as accounts in the database outline. This is true even if you have customized the default calculation script.

➤ To perform a default calculation, use any of the following methods:

Tool	Topic	Location
Administration Services	Calculating Block Storage Databases	*Essbase Administration Services Online Help*
MaxL	**execute calculation**	*Technical Reference*
ESSCMD	CALCDEFAULT	*Technical Reference*

➤ To enable two-pass calculation, use any of the following methods:

Tool	Topic	Location
Administration Services	Using Two-Pass on a Default Calculation	*Essbase Administration Services Online Help*
MaxL	**alter database**	*Technical Reference*
ESSCMD	SETDBSTATE	*Technical Reference*

Creating Calculation Scripts for Two-Pass and Intelligent Calculation

Use these methods to create calculation scripts to perform two-pass calculations with Intelligent Calculation, so that the calculation is accurate and as fast as possible:

- Before the calculation script command that recalculates a two-pass formula, add the SET UPDATECALC OFF command to disable Intelligent Calculation. If you have Intelligent Calculation enabled (the default), Analytic Services calculates only the data blocks that are not marked as clean, but when you perform a default calculation of the database with Intelligent Calculation enabled, all data blocks are marked as clean, so Analytic Services does not perform the two-pass formula recalculation.
- When you use a calculation script, Analytic Services does not automatically recalculate two-pass formulas. Use the CALC TWOPASS command.
- If you have changed the default calculation from the default CALC ALL and Intelligent Calculation is enabled, the data blocks may not be marked as clean after the first calculation. For more information, see Chapter 55, "Optimizing with Intelligent Calculation." You can check the default calculation setting by selecting the database, right-clicking and selecting Set > Default calculation in Administration Services.

To obtain the performance benefits of Intelligent Calculation when performing the first, full calculation of the database, use one of these methods, depending on the calculation needs and outline structure:

- "Intelligent Calculation with a Large Index" on page 1213
- "Intelligent Calculation with a Small Index" on page 1215
- "Intelligent Calculation Turned Off for a Two-Pass Formula" on page 1216

These three options all use the following example situation:

The outline has a dimension tagged as accounts, and it is a dense dimension. You want to calculate sales for each product as a percentage of sales for all products. Assume this formula should calculate the dimension:

```
Sales % Sales -> Product
```

When Analytic Services calculates the data block for each product, it has not yet calculated the value Sales->Product, so the results for the sales of each product as a percentage of total sales are incorrect.

Intelligent Calculation with a Large Index

If the index is quite large and you want the benefit of using Intelligent Calculation, you can use any of the following options for the best performance:

- "Use a Calculation Script" on page 1213
- "Use a Calculation Script and the Two-Pass Tag" on page 1214
- "Use a Client and a Calculation Script" on page 1214

All three of these options perform the same tasks:

1. Enable Intelligent Calculation.
2. Calculate the full database and marks the data blocks as clean.
3. Disable Intelligent Calculation.
4. Mark the recalculated blocks as clean, even though this calculation is a partial calculation of the database. If you do not use the command SET CLEARUPDATESTATUS AFTER, Analytic Services marks data blocks as clean only after a full calculation of the database.
5. Analytic Services cycles through the database calculating only the formula for the relevant member (Share of Sales in our example), or calculating all formulas tagged as two-pass in the database outline.

Use a Calculation Script

Use this model to create a calculation script that performs a full calculation of the database with Intelligent Calculation enabled:

```
SET UPDATECALC ON;
CALC ALL;
SET UPDATECALC OFF;
SET CLEARUPDATESTATUS AFTER;
"Share of Sales" = Sales % Sales -> Product;
```

Use a Calculation Script and the Two-Pass Tag

Use this procedure to tag a member as two-pass, and use a calculation script to calculate first the full database, then the two-pass member:

1. Place a formula in the database outline and tag it as two-pass.
2. Place the formula on the appropriate member in the dimension tagged as accounts, in our example, Share of Sales.
3. Create a calculation script that performs a full database calculation and then a two-pass calculation:

```
SET UPDATECALC ON;
CALC ALL;
SET UPDATECALC OFF;
SET CLEARUPDATESTATUS AFTER;
CALC TWOPASS;
```

Use a Client and a Calculation Script

Use this procedure to perform a default calculation from a client and then use a calculation script to perform the formula calculation:

1. Enable Intelligent Calculation if this default has been changed.
2. Perform a full calculation, using any of the tools listed in Table 83 on page 1215.
3. Use a calculation script similar to this example to disable Intelligent Calculation and calculate the formula:

```
SET UPDATECALC OFF;
SET CLEARUPDATESTATUS AFTER;
"Share of Sales" = Sales % Sales -> Product;
```

or:

```
SET UPDATECALC OFF;
SET CLEARUPDATESTATUS AFTER;
CALC TWOPASS;
```

Table 83: Methods for Performing a Full Calculation

Tool	Topic	Location
Administration Services	Calculating Databases	*Essbase Administration Services Online Help*
MaxL	**execute calculation**	*Technical Reference*
ESSCMD	CALCDEFAULT	*Technical Reference*

For a comprehensive discussion of Intelligent Calculation, see Chapter 55, "Optimizing with Intelligent Calculation."

For detailed information on developing formulas and calculation scripts, see Chapter 22, "Developing Formulas" and Chapter 27, "Developing Calculation Scripts."

Intelligent Calculation with a Small Index

If the index is small and you want the benefit of using Intelligent Calculation, use this procedure:

1. Create a calculation script to calculate the database, but tell Analytic Services not to mark the calculated data blocks as clean
2. Mark all data blocks as clean, and do not recalculate the data blocks.

```
SET CLEARUPDATESTATUS OFF;
CALC ALL;
CALC TWOPASS;
SET CLEARUPDATESTATUS ONLY;
CALC ALL;
```

With the example script, Analytic Services performs these tasks:

1. The SET CLEARUPDATESTATUS OFF command tells Analytic Services not to mark the calculated data blocks as clean.
2. The first CALC ALL command causes Analytic Services to cycle through the database calculating all dirty data blocks. Analytic Services does not mark the calculated data blocks as clean. Analytic Services does not automatically recalculate the formulas tagged as two-pass in the database outline.

3. The CALC TWOPASS command causes Analytic Services to cycle through the database recalculating the formulas that are tagged as two-pass in the dimension tagged as accounts in the database outline. Analytic Services recalculates the formulas because the required data blocks are not marked as clean by the previous CALC ALL. Analytic Services does not mark the recalculated data blocks as clean.
4. The SET CLEARUPDATESTATUS ONLY command tells Analytic Services to mark the data blocks as clean but not to calculate the data blocks. This command disables calculation.
5. The last CALC ALL command causes Analytic Services to cycle through the database and mark all the data blocks as clean. Analytic Services searches through the index and marks the data blocks as clean. It does not calculate the data blocks.

Intelligent Calculation Turned Off for a Two-Pass Formula

Create a calculation script that performs these tasks:

1. Disables Intelligent Calculation.
2. Performs a full calculation.
3. Repeats the two-pass formula:

```
SET UPDATECALC OFF;
CALC ALL;
"Share of Sales" = Sales % Sales -> Product;
```

Choosing Between Member Set Functions and Performance

Queries and calculations which reference a member that has been tagged as Dynamic Calc or Dynamic Calc and Store may be significantly slower than queries and calculations involving the same members, if the member has formulas involving any of these functions:

- @CURRMBR
- @PARENT
- @SPARENTVAL

- @ANCEST
- @SANCESTVAL

If you are experiencing slow performance, you may wish to either remove the dynamic calculation tag or remove these functions from the attached formula.

Consolidating #MISSING Values

If no data value exists for a combination of dimension members, Analytic Services gives the combination a value of #MISSING. Analytic Services treats #MISSING values and zero (0) values differently.

Understanding #MISSING calculation

This table shows how Analytic Services calculates #MISSING values. In this table, X represents any number:

Table 84: How Analytic Services Treats #MISSING Values (Continued)

Calculation/Operation	Result
X + #MISSING	X
X – #MISSING #MISSING – X	X -X
X * #MISSING	#MISSING
X / #MISSING #MISSING / X X / 0	#MISSING #MISSING #MISSING
X % #MISSING #MISSING % X X % 0	#MISSING #MISSING #MISSING
X == #MISSING	FALSE, unless X is #MISSING
X != #MISSING X <> #MISSING	TRUE, unless X is #MISSING TRUE, unless X is #MISSING
(X <= #MISSING)	(X <= 0)
(X >= #MISSING)	(X >= 0) or (X == #MISSING)

Table 84: How Analytic Services Treats #MISSING Values (Continued)

Calculation/Operation	Result
(X > #MISSING)	(X > 0)
(X < #MISSING)	(X < 0)
X AND #MISSING: Y AND #MISSING, where Y represents any nonzero value	#MISSING
0 AND #MISSING #MISSING AND #MISSING	0 #MISSING
X OR #MISSING: Y OR #MISSING, where Y represents any nonzero value 0 OR #MISSING #MISSING OR #MISSING	 1 #MISSING #MISSING
IF (#MISSING)	IF (0)
f (#MISSING)	#MISSING for any Analytic Services function of one variable
f (X)	#MISSING for any X not in the domain of *f* and any Analytic Services function of more than one variable (except where specifically noted)

By default, Analytic Services does not roll up #MISSING values. However, if you always load data at level 0 and never at parent levels, then you should enable the setting for consolidating #MISSING values. Use of this setting provides a calculation performance improvement of between 1% and 30%. The performance improvement varies, depending on database size and configuration.

CAUTION: The default, not consolidating #MISSING values, must be in effect if you load data at parent, rather than child, levels, if any child member combinations have #MISSING values. If all child member combinations have any other values, including zero (0), then Analytic Services rolls up the child values and overwrites the parent values correctly, so you can safely change the default.

Changing Consolidation for Performance

To consolidate, enable the setting for consolidating #MISSING values by using one of the methods described above. The degree of performance improvement you achieve depends on the ratio between upper level blocks and input blocks in the database.

➤ To change the way Analytic Services consolidates #MISSING values, use any of the following methods:

Tool	Topic	Location
Administration Services	Aggregating Missing Values During Calculation	*Essbase Administration Services Online Help*
Calculation script	SET AGGMISSG	*Technical Reference*
MaxL	**alter database**	*Technical Reference*
ESSCMD	SETDBSTATEITEM	*Technical Reference*

Note: If you enable the setting for consolidating #MISSING values, the cell calculation order within a data block changes. For more information, see "Cell Calculation Order" on page 527.

When the setting for consolidating #MISSING values is disabled, note that the performance overhead is particularly high in the following two situations:

- When you have a low ratio of calculated data blocks to input data blocks
- When you load many data values at parent levels on sparse dimensions; for example, in the Sample Basic database, if you load many data values into East in a sparse Market dimension

In these situations, the performance overhead is between 10% and 30%. If calculation performance is critical, you may want to reconsider the database configuration or reconsider how you load data.

For a detailed explanation of how Analytic Services calculates #MISSING values, see "Consolidating #MISSING Values" on page 1217.

Removing #MISSING Blocks

CLEARDATA changes the value of cells in a block to #MISSING. It does not remove the data blocks. These extra blocks can slow performance.

If the #MISSING blocks are slowing performance, perform either of these tasks:

- Use the CLEARBLOCK command to remove the data blocks.
- Export the data and import it again. For a review of methods, see "Exporting Data" on page 1083 and "Reloading Exported Data" on page 1084.

Identifying Additional Calculation Optimization Issues

The relationship between calculation and performance is also described in the following chapters:

- In Chapter 25, "Dynamically Calculating Data Values," see the following topics:
  - "Benefitting from Dynamic Calculation" on page 545
  - "Choosing Between Dynamic Calc and Dynamic Calc and Store" on page 550
  - "Reducing the Impact on Retrieval Time" on page 558
  - "Dynamically Calculating Data in Partitions" on page 565
- In Chapter 27, "Developing Calculation Scripts," see the following topics:
  - "Specifying Global Settings for a Database Calculation" on page 587
  - "Writing Calculation Scripts for Partitions" on page 602

For the relationship of two-pass calculation and the SET CLEARUPDATESTATUS command, see the *Technical Reference*.

When you convert currencies using the CCONV command, the resulting data blocks are marked as *dirty* for the purposes of Intelligent Calculation. This means that Analytic Services recalculates all the converted blocks when you recalculate the database. For a comprehensive discussion of Intelligent Calculation, see Chapter 55, "Optimizing with Intelligent Calculation."

Chapter

55

Optimizing with Intelligent Calculation

This chapter provides information on how to use Intelligent Calculation to optimize the performance of Analytic Services calculations. This chapter includes the following sections:

For information on using other methods to optimize database calculations, see the following chapters:

- Chapter 13, "Designing Partitioned Applications"
- Chapter 25, "Dynamically Calculating Data Values"
- Chapter 54, "Optimizing Calculations"

Introducing Intelligent Calculation

When you do a full calculation of a database, Analytic Services tracks which data blocks it has calculated. If you then load a subset of data, on subsequent calculations, you can choose to calculate only those data blocks that Analytic Services has not yet calculated but need calculation, and those calculated blocks that require recalculation because of the new data. This process is called Intelligent Calculation.

By default, Intelligent Calculation is turned on. You can change this default setting in the `essbase.cfg` file. For more information about the `essbase.cfg` file, see the *Technical Reference*.

You can also turn Intelligent Calculation on or off in a calculation script. For a brief explanation, see "Turning Intelligent Calculation On and Off" on page 1225.

Benefits of Intelligent Calculation

Intelligent Calculation is designed to provide significant calculation performance benefits for these types of calculations:

- For a full calculation of a database (CALC ALL)
- For a calculation script that calculates all members in one CALC DIM command.
- For database calculations that cannot use Intelligent Calculation for the full calculation, you may be able to use Intelligent Calculation for part of the calculation.

  For example, consider a case in which you calculate a database by doing a default consolidation and then an allocation of data. To significantly improve calculation performance in this case, enable Intelligent Calculation for the default consolidation and then disable Intelligent Calculation for the allocation.

  Assuming that Intelligent Calculation is turned on (the default), create a calculation script to perform these steps for a partial Intelligent Calculation:

  a. Enable Intelligent Calculation if the default has been changed
  b. Use CALC ALL to calculate the database
  c. Use the SET UPDATECALC command to disable Intelligent Calculation for the next step
  d. Allocate data
  e. Optionally, enable Intelligent Calculation again

Intelligent Calculation and Data Block Status

To provide Intelligent Calculation, Analytic Services checks the status of the data blocks in a database. Data blocks have a calculation status of either clean or dirty. Analytic Services marks a data block as clean after certain calculations.

When Intelligent Calculation is enabled, Analytic Services calculates only dirty blocks and their dependent parents. Disabling Intelligent Calculation means that Analytic Services calculates all data blocks, regardless of whether they are marked as clean or dirty.

Use these topics to understand clean and dirty status, and to learn how to manage clean and dirty status for Intelligent Calculation:

- "Marking Blocks As Clean" on page 1223
- "Marking Blocks as Dirty" on page 1224
- "Maintaining Clean and Dirty Status" on page 1224

Marking Blocks As Clean

Analytic Services marks data blocks as clean in these types of calculations:

- A full calculation (CALC ALL) of a database, the default calculation for a database.
- A calculation script that calculates all the dimensions in one CALC DIM statement. For example, the following calculation script calculates all members in the Sample Basic database:

  ```
  CALC DIM(Measures, Product, Market, Year, Scenario);
  ```

  Compare this calculation script to a calculation script that calculates all the members with two CALC DIM statements:

  ```
  CALC DIM(Measures, Product);
  CALC DIM(Market, Year, Scenario);
  ```

  Using two CALC DIM statements causes Analytic Services to do at least two calculation passes through the database. In this calculation, Analytic Services does not, by default, mark the data blocks as clean. Because Intelligent Calculation depends on accurate clean and dirty status, you must manage these markers carefully. For an example, see "Maintaining Clean and Dirty Status" on page 1224.

Analytic Services marks calculated data blocks as clean only in situations described above, unless you use the SET CLEARUPDATESTATUS command in a calculation script. For a comprehensive discussion of using the command, see "Using the SET CLEARUPDATESTATUS Command" on page 1227.

Marking Blocks as Dirty

Analytic Services marks a data block as dirty in the following situations:

- Calculating the data block for a partial calculation of the database only if SET CLEARUPDATESTATUS AFTER is not part of the partial calculation command in the calculation script
- Loading data into the data block
- Restructuring the database (for example, by adding a member to a dense dimension)
- Copying data to the data block, for example using DATACOPY

Maintaining Clean and Dirty Status

If you want to use Intelligent Calculation when calculating a subset of a database or when performing multiple calculation passes through a database, consider carefully the implications of how Analytic Services marks data blocks as clean. When using Intelligent Calculation, you must accurately maintain the clean and dirty status of the data blocks to ensure that Analytic Services recalculates the database as efficiently as possible.

For example, when you calculate a subset of a database, the newly calculated data blocks are not marked as clean by default. You can ensure that the newly calculated blocks are marked as clean by using the SET CLEARUPDATESTATUS AFTER command in a calculation script. To ensure accurate calculation results, review the information in "Using the SET CLEARUPDATESTATUS Command" on page 1227 and the *Technical Reference* (Calculation Commands List: CLEARUPDATESTATUS) before creating the calculation script.

Limitations of Intelligent Calculation

Consider the following limitations when using Intelligent Calculation:

- Intelligent Calculation works on a data block level and not on a cell level. For example, if you load a data value into one cell of a data block, the whole data block is marked as dirty.

- Changing a formula on the database outline or changing an accounts property on the database outline does not cause Analytic Services to restructure the database. Therefore, Analytic Services does not mark the affected blocks as dirty. You must recalculate the appropriate data blocks. For a review of methods, see "Changing Formulas and Accounts Properties" on page 1240.
- Whenever possible, Analytic Services calculates formulas that are tagged as two pass and in the dimension tagged as accounts as part of the main calculation of a database. However, you may need to use a calculation script to calculate some formulas twice. When you use a calculation script, disable Intelligent Calculation before recalculating formulas.
- When SET CREATENONMISSINGBLK is set to ON in a calculation script, Intelligent Calculation is turned off and affected blocks are calculated, regardless if they are marked clean or dirty.

Using Intelligent Calculation

This section provides information on turning Intelligent Calculation on and off and on using Intelligent Calculation with different types of calculations:

Turning Intelligent Calculation On and Off

By default, Intelligent Calculation is turned on. You can change the default by using the UPDATECALC setting in the `essbase.cfg` file.

You can turn Intelligent Calculation on and off for the duration of a calculation script by using the SET UPDATECALC command in a calculation script. Enabling Intelligent Calculation means that Analytic Services calculates only dirty blocks and their dependent parents. Disabling Intelligent Calculation means that Analytic Services calculates all data blocks, regardless of whether they are marked as clean or dirty.

For detailed information on these commands and on the `essbase.cfg` file, see the *Technical Reference*.

Using Intelligent Calculation for a Default, Full Calculation

Intelligent Calculation provides significant performance benefits when you do a full calculation (CALC ALL) of a database. If you do a full calculation of a database, leave Intelligent Calculation turned on (the default) to take advantage of the performance benefits that it provides.

Unless you have changed the default, a full calculation (CALC ALL) is the default calculation for a database.

➤ To check the current calculation setting, see "Setting the Default Calculation" in the *Essbase Administration Services Online Help*.

CAUTION: When using Intelligent Calculation, note the information in "Limitations of Intelligent Calculation" on page 1224.

Calculating for the First Time

When you do a full calculation of a database for the first time, Analytic Services calculates every existing block. The performance is the same whether you have Intelligent Calculation turned on or off.

Recalculating

When you do a full recalculation of a database with Intelligent Calculation turned on, Analytic Services checks each block to see if it is marked as clean or dirty. For an explanation of clean and dirty, see "Intelligent Calculation and Data Block Status" on page 1222.

Checking the data blocks has a 5% to 10% performance overhead. During most recalculations, this small performance overhead is insignificant when compared to the performance gained by enabling Intelligent Calculation.

However, if you recalculate a database in which more than approximately 80% of the values have changed, the overhead of Intelligent Calculation may outweigh the benefits. In this case, disable Intelligent Calculation.

Using Intelligent Calculation for a Calculation Script, Partial Calculation

Analytic Services marks a data block as clean when it calculates the data block on a full calculation (CALC ALL) or when it calculates all dimensions in one CALC DIM command. For a description of the types of calculations for which data blocks are marked clean, see "Marking Blocks As Clean" on page 1223.

In any other calculations, Analytic Services does not mark calculated data blocks as clean, unless you use the SET CLEARUPDATESTATUS command in a calculation script. For example, if you calculate a subset of a database or calculate a database in two calculation passes, Analytic Services does not mark the calculated blocks as clean, unless you use the SET CLEARUPDATESTATUS command.

The following calculation scripts do not cause Analytic Services to mark the calculated data blocks as clean:

```
FIX("New York")
CALC DIM(Product, Measures);
ENDFIX
CALC DIM(Measures, Product);
CALC DIM(Market, Year, Scenario);
```

Be sure to use SET CLEARUPDATESTATUS to avoid unnecessary recalculations.

Using the SET CLEARUPDATESTATUS Command

In some cases, Analytic Services does not mark calculated blocks as clean; for example, if you calculate a subset of a database or calculate a database in two calculation passes. To manually mark data blocks as clean for purposes of Intelligent Calculation, use the SET CLEARUPDATESTATUS command in a calculation script.

Use these sections to understand the command SET CLEARUPDATESTATUS, choose a setting, and for instructions about how to use the command:

- "Understanding SET CLEARUPDATESTATUS" on page 1228
- "Choosing a SET CLEARUPDATESTATUS Setting" on page 1229
- "Reviewing Examples That Use SET CLEARUPDATESTATUS" on page 1229

For a comprehensive discussion of the relationship between Intelligent Calculation and data block status, see "Intelligent Calculation and Data Block Status" on page 1222.

Understanding SET CLEARUPDATESTATUS

The SET CLEARUPDATESTATUS command has three parameters—AFTER, ONLY, and OFF.

- SET CLEARUPDATESTATUS AFTER;

  Analytic Services marks calculated data blocks as clean, even if it is calculating a subset of a database.

- SET CLEARUPDATESTATUS ONLY;

  Analytic Services marks the specified data blocks as clean but does not calculate the data blocks. This parameter provides the same result as AFTER, but without calculation.

- SET CLEARUPDATESTATUS OFF;

  Analytic Services calculates the data blocks but does not mark the calculated data blocks as clean. Data blocks are not marked as clean, even on a full calculation (CALC ALL) of a database. The existing clean or dirty status of the calculated data blocks remains unchanged.

Choosing a SET CLEARUPDATESTATUS Setting

When you use the SET CLEARUPDATESTATUS command to mark calculated data blocks as clean, be aware of these recommendations before selecting the parameter (AFTER, ONLY, OFF):

- Only calculated data blocks are marked as clean.
- Do not use the SET CLEARUPDATESTATUS AFTER command with concurrent calculations unless you are certain that the concurrent calculations do not need to calculate the same data block or blocks. If concurrent calculations attempt to calculate the same data blocks, with Intelligent Calculation enabled, Analytic Services does not recalculate the data blocks if the data blocks are already marked as clean by the other concurrent calculation. For an example, see "Handling Concurrent Calculations" on page 1233.
- When Analytic Services calculates data blocks on a first calculation pass through a database, it marks the data blocks as clean. If you try to calculate the same data blocks on a subsequent pass with Intelligent Calculation enabled, Analytic Services does not recalculate the data blocks because they are already marked as clean.

Reviewing Examples That Use SET CLEARUPDATESTATUS

Assume a scenario using the Sample Basic database:

- The sparse dimensions are Market and Product.
- New York is a member on the sparse Market dimension.
- Intelligent Calculation is turned on (the default).

These three examples show different ways of using SET CLEARUPDATESTATUS:

- "Example 1: CLEARUPDATESTATUS AFTER" on page 1230
- "Example 2: CLEARUPDATESTATUS ONLY" on page 1230
- "Example 3: CLEARUPDATESTATUS OFF" on page 1230

Example 1: CLEARUPDATESTATUS AFTER

```
SET CLEARUPDATESTATUS AFTER;
FIX(“New York”)
CALC DIM(Product);
ENDFIX
```

In this example, Analytic Services searches for dirty parent data blocks for New York (for example New York -> Colas, in which Colas is a parent member). It calculates these dirty blocks and marks them as clean. (The calculation is based on the Product dimension.) Analytic Services does not mark the level 0 data blocks as clean because they are not calculated. For information on level 0 blocks, see Chapter 24, “Defining Calculation Order.”

Example 2: CLEARUPDATESTATUS ONLY

```
SET CLEARUPDATESTATUS ONLY;
FIX(“New York”)
CALC DIM(Product);
ENDFIX
```

Analytic Services searches for dirty parent data blocks for New York (for example New York -> Colas, in which Colas is a parent member on the Product dimension). Analytic Services marks the dirty parent data blocks as clean, but does not calculate the data blocks. Analytic Services does not mark the level 0 data blocks as clean because they are not calculated. For example, if New York -> 100-10 (a level 0 block) is dirty, it remains dirty.

Example 3: CLEARUPDATESTATUS OFF

```
SET CLEARUPDATESTATUS OFF;
CALC ALL;
CALC TWOPASS;
SET CLEARUPDATESTATUS ONLY;
CALC ALL;
```

In this example, Analytic Services first calculates all the dirty data blocks in the database. The calculated data blocks remain dirty. Analytic Services does *not* mark them as clean.

Analytic Services then calculates the members tagged as two pass that are in the dimension tagged as accounts. Because the data blocks are still marked as dirty, Analytic Services recalculates them. Again, it does not mark the calculated data blocks as clean.

Analytic Services then searches for all the dirty blocks in the database and marks them as clean. It does *not* calculate the blocks, even though a CALC ALL command is used.

Calculating Data Blocks

Analytic Services creates a data block for each unique combination of sparse dimension members, provided that at least one data value exists for the combination. Each data block represents all dense dimension member values for that unique combination of sparse dimension members.

For example, in the Sample Basic database, the Market and Product dimensions are sparse. Therefore, the data block New York -> Colas represents all the member values on the Year, Measures, and Scenario dimensions for the sparse combination New York -> Colas.

These sections provide information about conditions that affect performance with Intelligent Calculation:

- "Calculating Dense Dimensions" on page 1231
- "Calculating Sparse Dimensions" on page 1232
- "Handling Concurrent Calculations" on page 1233
- "Understanding Multiple-Pass Calculations" on page 1234

These sections assumes that you are familiar with the concepts of upper level, level 0, and input data blocks. For explanations of these terms, to describe roles and relationships, and for an explanation of how Analytic Services creates data blocks, see "Data Storage in Data Blocks" on page 516.

Calculating Dense Dimensions

When you calculate a dense dimension and do not use a FIX command, Analytic Services calculates at least some of the data values in every data block in the database. For example, the following calculation script is based on the Sample Basic database:

```
SET CLEARUPDATESTATUS AFTER;
CALC DIM(Year);
```

This script calculates the Year dimension, which is a dense dimension. Because Year is dense, every data block in the database includes members of the Year dimension. Therefore, Analytic Services calculates data values in every data block. Because the script uses the SET CLEARUPDATESTATUS AFTER command, Analytic Services marks all the data blocks as clean.

Calculating Sparse Dimensions

When you calculate a sparse dimension, Analytic Services may not need to calculate every data block in the database. For example, the following calculation script is based on Sample Basic:

```
SET CLEARUPDATESTATUS AFTER;
CALC DIM(Product);
```

This script calculates the Product dimension, which is a sparse dimension. Because Product is sparse, a data block exists for each member on the Product dimension. For example, one data block exists for New York -> Colas and another for New York -> 100-10.

Level 0 Effects

The data block New York -> 100-10 is a level 0 block, it does not represent a parent member on either sparse dimension (Market or Product). The data values for New York -> 100-10 are input values; they are loaded into the database. Therefore, Analytic Services does not need to calculate this data block. Nor does Analytic Services mark the data block for New York -> 100-10 as clean, even though the script uses the SET CLEARUPDATESTATUS AFTER command.

Note: Analytic Services does calculate level 0 data blocks if a corresponding sparse, level 0 member has a formula applied to it.

If you load data into a database, the level 0 data blocks into which you load data are marked as dirty. If you subsequently calculate only a sparse dimension or dimensions, the level 0 blocks remain dirty, because Analytic Services does not calculate them. Therefore, when you recalculate only a sparse dimension or dimensions, Analytic Services recalculates all upper-level data blocks because the upper-level blocks are marked as dirty if their child blocks are dirty, even though the upper level blocks were originally clean.

Upper Level Effects

Colas is a parent level member on the Product dimension. Analytic Services needs to calculate values for Colas, so Analytic Services calculates this data block. Because the script uses the SET CLEARUPDATESTATUS AFTER command, Analytic Services marks the data block as clean.

When Analytic Services calculates a sparse dimension, it recalculates an upper level data block if the block is dependent on one or more dirty child blocks.

Unnecessary Calculation

You can avoid unnecessary calculation by ensuring that you calculate at least one dense dimension. When you calculate a dense dimension and do not use the FIX command, data values are calculated in every data block, including the level 0 blocks. So the level 0 blocks are marked as clean.

Handling Concurrent Calculations

If concurrent calculations attempt to calculate the same data blocks and Intelligent Calculation is turned on, Analytic Services may not recalculate the data blocks because they are already marked as clean.

Do not use the SET CLEARUPDATESTATUS AFTER command with concurrent calculations unless you are certain that the concurrent calculations do not calculate the same data block or blocks.

Consider the following example, which is based on the Sample Basic database. Actual and Budget are members of the dense Scenario dimension. Because Scenario is dense, each data block in the database contains both Actual and Budget values.

```
SET CLEARUPDATESTATUS AFTER;
FIX(“New York”, Actual)
CALC DIM(Product, Year);
ENDFIX
```

If User One runs the above calculation script, Analytic Services calculates the Actual values for all data blocks that represent New York. Analytic Services marks the calculated data blocks as clean, even though not all the data values in each calculated block have been calculated. For example, the Budget values have not yet been calculated.

```
SET CLEARUPDATESTATUS AFTER;
FIX(“New York”, Budget)
CALC DIM(Product, Year);
ENDFIX
```

If User Two runs the calculation script above to calculate the Budget values for New York, Analytic Services does not recalculate the specified data blocks, because they are already marked as clean. The calculation results for Budget are not correct.

One way to solve this problem is to make the Scenario dimension sparse; then the Actual and Budget values are in different data blocks, for example, New York -> Colas -> Actual and New York -> Colas -> Budget. In this case, the second calculation script correctly calculates Budget data block.

Understanding Multiple-Pass Calculations

Whenever possible, Analytic Services calculates a database in one calculation pass through the database. For an explanation of why and how multiple calculation passes are performed, see “Calculation Passes” on page 536.

When you use a calculation script to calculate a database, the number of calculation passes that Analytic Services performs depends upon the calculation script. For detailed information about the relationship between calculation passes and Intelligent Calculation, see “Intelligent Calculation and Data Block Status” on page 1222. For information about grouping formulas and calculations, see “Grouping Formulas and Calculations” on page 593.

For example, assume Analytic Services calculates data blocks on a first calculation pass through a database and then marks them as clean. If you then attempt to calculate the same data blocks on a subsequent pass and Intelligent Calculation enabled, Analytic Services does not recalculate the data blocks because they are already marked as clean.

Reviewing Examples and Solutions for Multiple-Pass Calculations

These examples describe situations in which you obtain incorrect calculation results, and provide a solution you can implement to obtain correct results:

The examples are based on the Sample Basic database and assume that Intelligent Calculation is turned on.

Example 1: Intelligent Calculation and Two Pass

This calculation script does a default calculation and then a two-pass calculation:

```
CALC ALL;
CALC TWOPASS;
```


Error

Analytic Services calculates the dirty data blocks in the database and marks all the data blocks as clean. Analytic Services then needs to recalculate the members tagged as two pass in the dimension tagged as accounts. However, Analytic Services does not recalculate the specified data blocks because they are already marked as clean. The calculation results are not correct.

Solution

You can calculate the correct results by disabling Intelligent Calculation for the two pass calculation.

Example 2: SET CLEARUPDATESTATUS and FIX

This calculation script calculates data values for New York. The calculation is based on the Product dimension:

```
SET CLEARUPDATESTATUS AFTER;
FIX(“New York”)
CALC DIM(Product);
ENDFIX
CALC TWOPASS;
```

Error

Analytic Services performs the following processes:

1. Analytic Services cycles through the database calculating the dirty data blocks that represent New York. The calculation is based on the Product dimension. Thus, Analytic Services calculates only the blocks that represent a parent member on the Product dimension (for example, New York -> Colas, New York -> Root Beer, and New York -> Fruit Soda), and then only calculates the aggregations and formulas for the Product dimension.
2. Because the SET CLEARUPDATESTATUS AFTER command is used, Analytic Services marks the calculated data blocks as clean, even though not all data values in each calculated block have been calculated.
3. Analytic Services should recalculate the members tagged as two pass in the dimension tagged as accounts. However, some of these data blocks are already marked as clean from the calculation in step 2. Analytic Services does not recalculate the data blocks that are already marked as clean. The calculation results are not correct.

Solution

You can calculate the correct results by disabling Intelligent Calculation for the two pass calculation.

Example 3: SET CLEARUPDATESTATUS and Two CALC DIM Commands

This calculation script bases the database calculation on the Product and Year dimensions. Because two CALC DIM commands are used, Analytic Services does two calculation passes through the database:

```
SET CLEARUPDATESTATUS AFTER;
CALC DIM(Product);
CALC DIM(Year);
```

Error

Analytic Services performs the following processes:

1. Analytic Services cycles through the database calculating the dirty data blocks. The calculation is based on the Product dimension, as in Example 2.
2. Because the SET CLEARUPDATESTATUS AFTER command is used, Analytic Services marks the calculated data blocks as clean, even though not all data values in each calculated block have been calculated.
3. Analytic Services should recalculate the data blocks. The recalculation is based on the Year dimension. However, as a result of the calculation in step 2, some of the data blocks are already marked as clean. Analytic Services does not recalculate the data blocks that are already marked as clean. The calculation results are not correct.

Solution

You can calculate the correct results by using one CALC DIM command to calculate both the Product and Year dimensions. Analytic Services then calculates both dimensions in one calculation pass through the database. The following calculation script calculates the correct results:

```
SET CLEARUPDATESTATUS AFTER;
CALC DIM(Product, Year);
```

Note: When you calculate several dimensions in one CALC DIM command, Analytic Services calculates the dimensions in the default calculation order and not in the order in which you list them in the command. For a description of default calculation order, see “Member Calculation Order” on page 517.

Example 4: Two Separate Calculation Scripts

This example calculates data values for New York but calculates based on two different dimensions using two separate calculation scripts. The first calculation script calculates the Product dimension:

```
SET CLEARUPDATESTATUS AFTER;
FIX(“New York”)
CALC DIM(Product);
ENDFIX
```

Analytic Services calculates the data blocks that include New York. The calculation is based on the Product dimension. Thus, Analytic Services calculates only the dirty blocks that include a parent member on the Product dimension (for example, New York -> Colas, New York -> Root Beer, and New York -> Fruit Soda), and even then only calculates the aggregations and formulas for the Product dimension.

Because of the CLEARUPDATESTATUS AFTER command, Analytic Services marks the calculated data blocks as clean, even though not all data values in each calculated block have been calculated.

The second calculation script calculates the Year dimension:

```
SET CLEARUPDATESTATUS AFTER;
FIX(“New York”)
CALC DIM(Year);
ENDFIX
```

Analytic Services calculates the data blocks that represent New York. The calculation is based on the Year dimension, which is a dense dimension. Thus, Analytic Services should calculate all data blocks that include New York, although within each block, Analytic Services calculates only the aggregations and formulas for the Year dimension.

Error

As a result of the first calculation, some of the data blocks for New York are already marked as clean. Analytic Services does not recalculate these data blocks with the second calculation script because the data blocks are already marked as clean. The calculation results are not correct.

Solution

You can calculate the correct results by telling Analytic Services *not* to mark the calculated data blocks as clean. The following calculation script calculates the correct results:

```
SET CLEARUPDATESTATUS OFF;
FIX(“New York”)
CALC DIM(Product);
ENDFIX
SET CLEARUPDATESTATUS AFTER;
FIX(“New York”)
CALC DIM(Year);
ENDFIX
```

The SET CLEARUPDATESTATUS OFF command. With it, Analytic Services calculates dirty data blocks, but does not to mark them as clean, unlike the SET CLEARUPDATESTATUS AFTER command.

This solution assumes that the data blocks are not already marked as clean from a previous partial calculation of the database.

You can ensure that all data blocks are calculated, irrespective of their clean or dirty status, by disabling Intelligent Calculation.

The following calculation script calculates all specified data blocks, irrespective of their clean or dirty status:

```
SET UPDATECALC OFF;
FIX(“New York”)
CALC DIM(Year, Product);
ENDFIX
```

Because you have not used the SET CLEARUPDATESTATUS AFTER command, Analytic Services does not mark calculated data blocks as clean.

Understanding the Effects of Intelligent Calculation

Using Intelligent Calculation may have implications for the way you administer a database. This section discusses the implications of each of the following actions:

- “Changing Formulas and Accounts Properties” on page 1240
- “Using Relationship and Financial Functions” on page 1240

- "Restructuring Databases" on page 1241
- "Copying and Clearing Data" on page 1241
- "Converting Currencies" on page 1241

Changing Formulas and Accounts Properties

Neither changing a formula in the database outline nor changing an accounts property in the database outline causes Analytic Services to restructure the database. Thus, data blocks affected by such a change are not marked as dirty. For example, if you change a time balance tag in the dimension tagged as accounts, Analytic Services does not restructure the database and does not mark the affected blocks as dirty.

When you subsequently run a default calculation with Intelligent Calculation turned on, the changes are not calculated. To recalculate the appropriate data blocks, use a calculation script to perform any of the following tasks:

- Disable Intelligent Calculation and calculate the member formula that has changed.
- Disable Intelligent Calculation and use the FIX command to calculate the appropriate subset of a database.
- Disable Intelligent Calculation and perform a default CALC ALL on a database.

Using Relationship and Financial Functions

If you use relationship functions (for example, @PRIOR or @NEXT) or financial functions (for example, @ACCUM, @NPV, or @INTEREST) in a formula on a sparse dimension or a dense dimension, Analytic Services always recalculates the data block that contains the formula.

For detailed information on specific relationship functions and financial functions, see the *Technical Reference*.

Restructuring Databases

When you restructure a database (for example, by adding a member to a dense dimension), all data blocks potentially need recalculating. Therefore, Analytic Services marks all data blocks as dirty. When you calculate the restructured database, all blocks are calculated.

Note: Changing a formula in the database outline or changing an accounts property in the database outline does not cause Analytic Services to restructure the database. You must recalculate the appropriate data blocks. For more information, see "Changing Formulas and Accounts Properties" on page 1240.

Copying and Clearing Data

When you copy values to a data block by using the DATACOPY command, the resulting data block is marked as dirty. Analytic Services calculates the block when you recalculate a database.

When you clear data values by using the CLEARDATA and CLEARBLOCK commands, Analytic Services clears all the blocks regardless of whether they are marked as clean or dirty.

Converting Currencies

When you convert currencies using the CCONV command, the resulting data blocks are marked as dirty. Analytic Services calculates all converted blocks when you recalculate a database.

Chapter 56

Optimizing Reports and Other Types of Retrieval

The time required to generate a report varies depending upon factors such as the size of the database you are reporting from, the number of queries included in the script, and the size of the report buffer.

This chapter describes ways to optimize the time required to generate reports, and other retrieval optimization issues:

If you are migrating from a previous version of Analytic Services, see the *Essbase Analytic Services Installation Guide* for information about system changes and enhancements.

Changing Buffer Size

Configurable variables specify the size of the buffers used for storing and sorting data extracted by retrievals. The buffer should be large enough to prevent unnecessary read and write activities.

These report variables are used in the conditional retrieval and data sorting commands:

- "Setting the Retrieval Buffer Size" on page 1244
- "Setting the Retrieval Sort Buffer Size" on page 1245

For a description of the Report Extractor process of retrieving data, see "Report Extractor" on page 661.

Setting the Retrieval Buffer Size

The database retrieval buffer is a server buffer, per database, that holds extracted row data cells before they are evaluated. Retrieval tools such as the Spreadsheet Add-in Retrieval Wizard and the Report Writer use this buffer.

When the retrieval buffer is full, the rows are processed, and the buffer is reused. If this buffer is too small, frequent reuse of the area can increase retrieval times. If this buffer is too large, too much memory may be used when concurrent users perform queries, resulting in increased retrieval times.

The following sections describe ways you can manage retrieval buffer sizes:

- "Manually Setting the Retrieval Buffer Size" on page 1244
- "Enabling Dynamic Retrieval Buffer Sizing" on page 1245

Manually Setting the Retrieval Buffer Size

Each database has a retrieval buffer setting, in kilobytes, that you can change. The default buffer size is set to 10 kilobytes. If you are increasing the size of the buffer, it is recommended that you do not exceed 100 kilobytes, although the size limit is set at 100,000 kilobytes.

➤ To manually set the retrieval buffer size, use any of the following methods:

Tool	Topic	Location
Administration Services	Setting the Size of Retrieval Buffers	*Essbase Administration Services Online Help*
MaxL	**alter database**	*Technical Reference*
ESSCMD	SETDBSTATEITEM	*Technical Reference*

Note: If an outline does not include Dynamic Calc, Dynamic Times Series, or attribute members, using the VLBREPORT configuration parameter to dynamically determine the size of the retrieval buffer overrides the database retrieval buffer size setting. See "Enabling Dynamic Retrieval Buffer Sizing" on page 1245.

Enabling Dynamic Retrieval Buffer Sizing

If a database has very large block size and retrievals include a large percentage of cells from each block across several blocks, consider setting the VLBREPORT option to TRUE in the Essbase configuration file `essbase.cfg`.

When the VLBREPORT setting is TRUE, Essbase internally determines an optimized retrieval buffer size for reports that access more than 20% of the cells in each block across several blocks. This setting takes effect only if the outline does not include Dynamic Calc, Dynamic Times Series, or attribute members. The VLBREPORT configuration setting overrides the manually set retrieval buffer size.

Setting VLBREPORT to TRUE can result in faster query response times for concurrent and serial queries at the cost of a slight increase in memory required for each query.

Setting the Retrieval Sort Buffer Size

The retrieval sort buffer size setting specifies the size, in kilobytes, of the server buffer that holds the data to be sorted during a Spreadsheet Add-in or Report Writer retrieval. If the sorting buffer is full, Analytic Services posts an error message.

You can adjust the buffer size on a per-database basis. The default buffer size is set to 10 kilobytes.

➤ To set the retrieval sort buffer size, use any of the following methods:

Tool	Topic	Location
Administration Services	Setting the Size of Retrieval Buffers	*Essbase Administration Services Online Help*
MaxL	**alter database**	*Technical Reference*
ESSCMD	SETDBSTATEITEM	*Technical Reference*

Setting Numeric Precision

The NUMERICPRECISION setting, used by the RESTRICT command, defines the number of precision digits the internal numerical comparison considers in the Report Extractor. If you have a precision setting greater than necessary for the data, the retrieval is slower than it could be. Identify the correct level of precision and then adjust NUMERICPRECISION accordingly.

This table lists the setting that you specify in the `essbase.cfg` file on the server to set the NUMERICPRECISION. If you change an `essbase.cfg` setting, restart Analytic Server to apply the change.

Table 85: Setting Messages in the Server Using essbase.cfg

Setting	Definition	For More Information
NUMERICPRECISION	An `essbase.cfg` setting that determines the number of precision digits used by Report Extractor.	*Technical Reference*

Generating Symmetric Reports

If report processing time is of primary importance, and you are using Report Writer, consider making all reports symmetric. Symmetric reports provide better processing performance than asymmetric reports, as the Report Extractor composes the member list based on all possible member combinations. A list of

this nature allows Report Extractor to create the list in one pass. With asymmetric reports, the Extractor must retrieve and process each block of possible member combinations separately

Figure 245: Symmetric Report Member Combinations Supporting One Pass

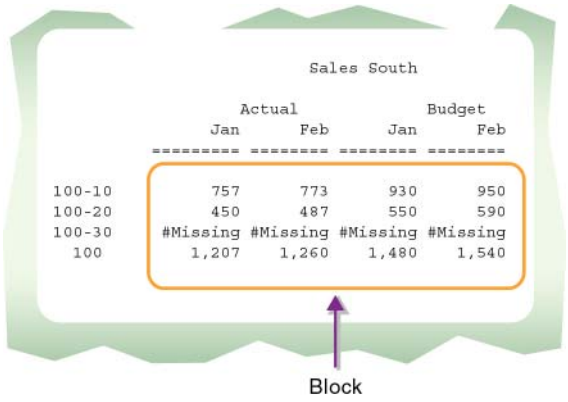
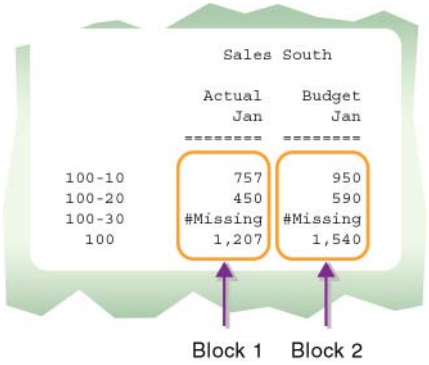


Figure 246: Asymmetric Report Member Combinations Requiring Multiple Passes



For a description of how the Report Extractor retrieves data, see “Report Extractor” on page 661.

Organizing Members to Optimize Data Extraction

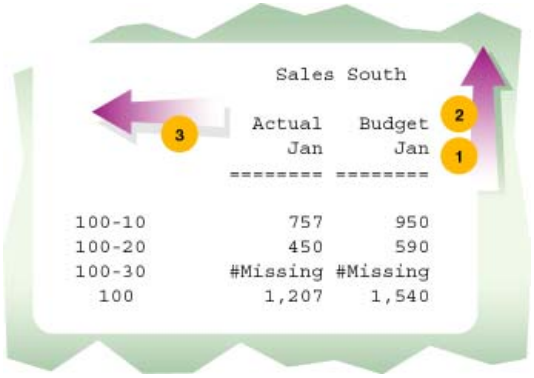
Report Extractor extracts data in a certain order for the Report Writer. If you do not require a formatted report and you are using Report Writer, you can reduce the time required to generate the report by using any of these strategies:

- Creating the report script in the same order as Report Extractor extracts data
- Grouping dense dimensions in columns and grouping sparse dimensions in rows

These strategies save the most time if used to create large production reports.

Report Extractor looks at data from bottom to top and right to left, starting from the bottom column member to the top column member and then proceeding from the innermost row member (right) to the outermost row member (left). Figure 247 illustrates the sequence in which the report is read.

Figure 247: How Report Extractor Examines Data



The column members come from dense dimensions, and the row members come from sparse dimensions. To reduce the time needed to extract data, group dense dimensions first, then group sparse dimensions in the same sequence as they are displayed in the outline.

When dense dimensions are nested in the report columns, Report Extractor examines each data block only once, thus improving performance time.

Attributes are sparse dimensions and are dynamically calculated. Hence, Analytic Services cannot use the sparse data extraction method when a report contains attribute dimensions.

Understanding Reports for Outlines That Contain Dynamic or Transparent Members

If you generate a report that accesses a database outline that contains Dynamic Calc and Store members, the first time that you generate the report takes longer than subsequent retrievals that access the same data block.

If you generate a report that accesses a database outline that contains Dynamic Calc or Dynamic Time Series members, Analytic Services calculates the member every time a report is generated, which increases the reporting time.

For a comprehensive discussion of dynamic calculation, see Chapter 25, “Dynamically Calculating Data Values.”

If you run a report that contains transparent members, the report takes longer to generate, as it must access more than one server to retrieve the required data.

Limiting LRO File Sizes

You can limit the size of files that users can link to a database. Limiting the size prevents a user from taking up too much of the server resources by storing extremely large objects. For more information, see “Limiting LRO File Sizes for Storage Conservation” on page 214.

Index

Symbols

Numerics

A

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

R

S

T

U

V

W

X

Z

Essbase Analytic Services

Release 7.1

Database Administrator's Guide

Volume IV: Creating, Calculating, and Managing Aggregate Storage Databases

Hyperion Solutions Corporation

Hyperion Solutions Corporation
1344 Crossman Avenue
Sunnyvale, California 94089

Printed in the U.S.A.

Contents

Chapter 57

Comparison of Aggregate and Block Storage

Essbase Analytic Services provides an aggregate storage kernel as a persistence mechanism for multidimensional databases. Aggregate storage databases enable dramatic improvements in both database aggregation time and dimensional scalability. The aggregate storage kernel is an alternative to the block storage kernel. Aggregate storage databases typically address read-only, “rack and stack” applications that have large dimensionality, such as the following applications:

- Customer analysis. Data is analyzed from any dimension and there are potentially millions of customers.
- Procurement analysis. Many products are tracked across many vendors.
- Logistics analysis. Near real-time updates of product shipments are provided.

A new sample application (ASOsamp), a data file, and a rules file are provided to demonstrate aggregate storage functionality.

Differences Between Aggregate and Block Storage

Aggregate storage applications differ from block storage applications in both concept and design. Additionally, aggregate storage applications have some limitations that do not apply to block storage applications. The following tables describe the differences between aggregate and block storage.

- Table 86, “Inherent Differences Between Aggregate Storage and Block Storage” on page 1288
- Table 87, “Outline Differences Between Aggregate Storage and Block Storage” on page 1289

- Table 88, "Calculation Differences Between Aggregate Storage and Block Storage" on page 1293
- Table 89, "Partitioning and Write Back Differences Between Aggregate Storage and Block Storage" on page 1293
- Table 90, "Data Load Differences Between Aggregate Storage and Block Storage" on page 1294
- Table 91, "Query Differences Between Aggregate Storage and Block Storage" on page 1294
- Table 92, "Features Supported by Aggregate and Block Storage" on page 1295

Table 86: Inherent Differences Between Aggregate Storage and Block Storage

Inherent Differences	Aggregate Storage	Block Storage
Storage kernel	Architecture that supports rapid aggregation, optimized to support high dimensionality and sparse data	Multiple blocks defined by dense and sparse dimensions and their members, optimized for financial applications
Physical storage definition	Through the Application Properties window, Tablespaces tab in Essbase Administration Services	Through the Database Properties window, Storage tab in Essbase Administration Services.
Database creation	Migrate a block storage outline or define after application creation **Note:** Do not use the file system to copy a block storage outline into an aggregate storage application. Use the migration wizard in Administration Services to Migrate the outline	Define after application creation
Databases supported per application	One	Several

Table 86: Inherent Differences Between Aggregate Storage and Block Storage(Continued)

Inherent Differences	Aggregate Storage	Block Storage
Application and database names	Names reserved for tablespaces, cannot be used as application or database names: • default • log • metadata • temp	No reserved names
Application and database information display	Displayed in the Application Properties window and the Database Properties window in Administration Services (Information not supported by or relevant to aggregate storage applications is not shown. For a description of aggregate storage specific information, see *Essbase Administration Services Online Help* for the Application Properties window and Database Properties window)	Displayed in the Application Properties window and the Database Properties window in Administration Services
Configuration settings (`Essbase.CFG`)	For a list of the settings that apply to aggregate storage databases, see the *Technical Reference*.	For a list of the settings that do not apply to block storage databases, see the *Technical Reference*.

Table 87: Outline Differences Between Aggregate Storage and Block Storage

Outline Functionality	Aggregate Storage	Block Storage
Dense or sparse dimension designation	Not relevant	Relevant

Table 87: Outline Differences Between Aggregate Storage and Block Storage (Continued)

Outline Functionality	Aggregate Storage	Block Storage
Accounts dimensions	Support with the following exceptions: • No time balance members • No two-pass calculation • No association of attribute dimensions (See also "Formulas" on page 1293.)	Full support
Non-accounts dimensions	Support with the following exceptions: • No member formulas • Support for only the + (addition) consolidation operator • Restrictions on label only members • No Dynamic Time Series members (See also "Member consolidation properties" on page 1290 and "Member storage types" on page 1291.)	Full support
Member consolidation properties	For non-accounts dimensions, support for only the + (addition) consolidation operator	For all dimensions, support for all consolidation properties

Table 87: Outline Differences Between Aggregate Storage and Block Storage (Continued)

Outline Functionality	Aggregate Storage	Block Storage
Member storage types	Support with the following exceptions: • No shared members • Never Share member not relevant • Dynamic Calc and Store not relevant • On non-accounts dimensions, two limitations if a member is label only: – All dimension members at the same level as the member must be label only – The parents of the member must be label only. **Note:** On accounts dimensions, ability to tag any member as label only **Note:** On conversion from a block storage database, attribute dimension members are tagged as Dynamic Calc. On standard dimension members Dynamic Calc tags are converted and tagged as stored members, which changes the Members Stored value on the Dimensions tab of the Database Properties window in Administration Services.	Support for all member storage types in all types of dimensions except attribute dimensions
Ragged hierarchies and hierarchies with more than 10 levels	Support, with possible performance impact	Support
Outline validation	• When database is started • When outline is saved • When block storage outline is converted to aggregate storage outline • When user requests	• When outline is saved • When user requests
Outline paging	Support	No support

Table 87: Outline Differences Between Aggregate Storage and Block Storage (Continued)

Outline Functionality	Aggregate Storage	Block Storage
Database restructure	The following actions cause Analytic Services to restructure the outline and clear all data: • Add, delete, or move a standard dimension member • Add, delete, or move a standard dimension • Add, delete, or move an attribute dimension • Add a formula to a level 0 member • Delete a formula from a level 0 member The following actions cause an outline restructure, but do not clear the data: • Add, delete, or move an attribute dimension member • Add or delete a formula from a non-level 0 member • Rename a member or dimension • Add or delete an alias • Add or delete member comments or extended member comments • Add or delete a user-defined attribute (UDA) • Associate or disassociate an attribute dimension	Several levels of restructure. See Chapter 52, "Optimizing Database Restructuring".

Table 88: Calculation Differences Between Aggregate Storage and Block Storage

Calculation Functionality	Aggregate Storage	Block Storage
Database calculation	Aggregation of the database, which can be predefined by defining aggregate views	Calculation script or outline consolidation
Formulas	Allowed with the following restrictions: • Only in the dimension tagged as accounts • Must be valid numeric value expressions written in MDX (cannot contain % operator, replace with expression: (*value1/value2*)*100) • No support for Analytic Services calculation functions	Allowed in all dimension types
Calculation scripts	Not supported	Supported
Attribute calculations dimension	Support for Sum	Support for Sum, Count, Min, Max, and Average
Calculation order	Not relevant (Predefined by the outline)	Defined by the user in the outline consolidation order or in a calculation script

Table 89: Partitioning and Write Back Differences Between Aggregate Storage and Block Storage

Partitioning and Write-Back Functionality	Aggregate Storage	Block Storage
Partitioning	Support with the following restrictions: • Transparent partitions only • Aggregate storage database as the source database • No outline synchronization	Support with no restrictions
User ability to change data (write back)	Transparent partition technique used to enable limited write back	Full support

Table 90: Data Load Differences Between Aggregate Storage and Block Storage

Data Load Functionality	Aggregate Storage	Block Storage
Cells loaded through data loads	Only level-0 cells whose values do not depend on formulas in the outline are loaded	Cells at all levels can be loaded
Update of database values	At the end of a data load, if an aggregation exists, the values in the aggregation are recalculated and updated.	No automatic update of values. To update data values you must execute all necessary calculation scripts.
Incremental data load	Data values are cleared each time the outline is changed structurally. Therefore, incremental data loads are supported only for outlines that do not change (for example, logistics analysis applications).	Outline changes do not automatically clear data values, even if a data source is used to both modify members and load values. Therefore, incremental data loads are supported for all outlines.

Table 91: Query Differences Between Aggregate Storage and Block Storage

Query Functionality	Aggregate Storage	Block Storage
Report Writer	Supported, except for commands related to sparsity and density of data	Fully supported
Spreadsheet Add-in	Supported, with limited ability to change data (write back) (See "User ability to change data (write back)" on page 1293.)	Fully supported
API	Supported	Supported
Export	Not supported, except for the MaxL **export data using report_file** grammar	Supported

Table 91: Query Differences Between Aggregate Storage and Block Storage

Query Functionality	Aggregate Storage	Block Storage
MDX queries	Supported	Supported
Queries on attribute members that are associated with non-level 0 members	Returns values for descendants of the non-level 0 member.	Returns #MISSING for descendants of the non-level 0 member.

Features Supported by Aggregate and Block Storage

Some features are not supported for aggregate storage. The following table describes the differences between aggregate and block storage.

Table 92: Features Supported by Aggregate and Block Storage

Features	Aggregate Storage	Block Storage
Aliases	Supported	Supported
Currency conversion	Not supported	Supported
Data mining	Not supported	Supported

Hybrid analysis	Support with the following restriction: queries that contain a relational member and an Analytic Services member with a formula in the same query are not supported.
	For example, if California is a relational member and the member Profit has a formula, the following report script returns an error:
	`Jan`
	`California`
	`Profit`
	`!`
		Supported

Linked reporting objects (LROs)	Not supported	Supported
Time balance reporting	Not supported	Supported

Table 92: Features Supported by Aggregate and Block Storage (Continued)

Features	Aggregate Storage	Block Storage
Triggers	After-update triggers supported	On-update triggers and after-update triggers supported
Unicode	Not supported	Supported
Variance reporting	Not supported	Supported

Chapter

58

Aggregate Storage Applications, Databases, and Outlines

This chapter provides information about creating aggregate storage applications, databases, and outlines and discusses the differences between aggregate storage databases and block storage databases in regard to the creation of applications, databases, and outlines. To use the information in these topics, you should be familiar with application, database, and outline concepts for block storage databases. For information about these concepts, see "Creating Applications and Databases" on page 125, "Creating and Changing Database Outlines" on page 139, "Setting Dimension and Member Properties" on page 153, and "Working with Attributes" on page 179.

Aggregate storage applications and databases and block storage applications and databases differ in both concept and design. Some block storage outline features do not apply to aggregate storage. For example, the concept of dense and sparse dimensions does not apply. Also, in aggregate storage outlines, formulas are allowed only within the dimension tagged as accounts and must be written in MDX syntax. For a full list of differences, see Chapter 57, "Comparison of Aggregate and Block Storage".

A new sample application (ASOsamp), a data file, and a rules file are provided to demonstrate aggregate storage functionality.

This chapter includes the following topics:

Workflow for Creating Aggregate Storage Applications

This topic provides a high-level workflow for creating an aggregate storage application.

1. Create an aggregate storage application, database, and outline. See "Creating Aggregate Storage Applications, Databases, and Outlines" on page 1299.
2. Use tablespaces to optimize data storage and retrieval. See "Managing Storage for Aggregate Storage Applications" on page 1343.
3. Specify the maximum size of the aggregate storage cache. See "Managing the Aggregate Storage Cache" on page 1345.
4. Load data into the aggregate storage database. A data load can be combined with a dimension build. See "Preparing Aggregate Storage Databases" on page 1325. You can preview a subset of the data in Essbase Administration Services. See "Previewing Data" in *Essbase Administration Services Online Help*.
5. Precalculate chosen aggregations to optimize retrieval time. See "Calculating Aggregate Storage Databases" on page 1333.
6. View database statistics. See "Viewing Aggregate Storage Statistics" in *Essbase Administration Services Online Help*.
7. If required, enable write-back by using the Aggregate Storage Partition Wizard. See "Using a Transparent Partition to Enable Write-Back for Aggregate Storage Databases" on page 1317.
8. View data using Hyperion tools (for example Essbase Spreadsheet Add-in) or third-party tools.

Creating Aggregate Storage Applications, Databases, and Outlines

You must create an aggregate storage application to contain an aggregate storage database. An aggregate storage application can contain only one database. You can create an aggregate storage application, database, and outline in the following ways:

- Convert a block storage outline to an aggregate storage outline, and create an aggregate storage application to contain the converted database and outline.

  Note: An aggregate storage outline cannot be converted to a block storage outline.

- Create an aggregate storage application and database. The aggregate storage outline is created automatically when you create the database.

For information on loading dimensions and members into an aggregate storage outline, see "Building Dimensions in Aggregate Storage Databases" on page 1325 and "Loading Data into Aggregate Storage Databases" on page 1328.

Aggregate storage application and database information differs from block storage information, and specific naming restrictions apply to aggregate storage applications and databases. For information on the differences, see "Inherent Differences Between Aggregate Storage and Block Storage" on page 1288.

➤ To convert a block storage outline to an aggregate storage outline, use any of the following methods:

Tool	Topic	Location
Administration Services	Aggregate Storage Outline Conversion Wizard	*Essbase Administration Services Online Help*
MaxL	**create outline**	*Technical Reference*

Note: Do not use the file system to copy a block storage outline into an aggregate storage application. Use the Aggregate Storage Outline Conversion Wizard in Essbase Administration Services to convert the outline.

➤ To create an aggregate storage application, use any of the following methods:

Tool	Topic	Location
Administration Services	Creating Applications	*Essbase Administration Services Online Help*
MaxL	**create application**	*Technical Reference*

➤ To create an aggregate storage database use any of the following methods:

Tool	Topic	Location
Administration Services	Creating Databases	*Essbase Administration Services Online Help*
MaxL	**create database**	*Technical Reference*

When creating aggregate storage applications, databases, and outlines, you need to consider differences between aggregate storage and block storage and issues specific to aggregate storage. For information about such differences and issues, see the following topics:

- "Outline Paging" on page 1300
- "Optimizing Outline Paging" on page 1304
- "Choosing an Accounts Dimension" on page 1307
- "Attribute Dimensions" on page 1308
- "Verifying Outlines" on page 1308

Outline Paging

Aggregate storage database outlines are pageable. This feature may significantly reduce memory usage for very large database outlines. For aggregate storage databases, Essbase Analytic Services preloads part of the database outline into memory. Then, during data retrieval, Analytic Services pages other parts of the outline into memory as required.

When you create an aggregate storage database, the outline is created in a pageable format. When you use the Aggregate Storage Outline Conversion Wizard to convert an existing block storage outline to aggregate storage, the outline is automatically converted to a pageable format.

Paging an outline into memory enables Analytic Services to handle very large outlines (for example, 10 million or more members), but potentially increases data retrieval time. You can customize outline paging to obtain the optimum balance between memory usage and data retrieval time. For configuration information, see "Optimizing Outline Paging" on page 1304.

Note: Aggregate storage databases that have pageable outlines contain memory pages, and therefore their outline files may be larger than binary, block storage database outline files.

Outline Paging Limits

The maximum size of a buildable outline (the number of members) depends on a number of factors:

- The available memory for Analytic Services
- The amount of memory in Analytic Services allocated for other uses
- The amount of memory required for each member (and aliases for each member)

Table 93 shows the amount of addressable memory available for Analytic Services for different operating systems.

Table 93: Addressable Memory Per Operating System

Operating System	Addressable Memory
Windows 2000, Windows 2003, Window XP	2 GB addressable memory 1.85 GB available to any application, including Analytic Services
Windows 2000 Advanced Server Windows 2003 Advanced Server	3 GB Requires a setting in the `boot.ini` file
AIXU	2 GB Up to 8 (256 MB) segments

Table 93: Addressable Memory Per Operating System (Continued)

Operating System	Addressable Memory
AIX 5.*x*	3.25 GB Up to 13 (256 MB) segments. Requires setting the LDR_CNTRL environment variable to: `0xD0000000@DSA`
HP-UX	2.9 GB Requires using the following command to set the addressable memory for the Essbase server process, ESSSVR: `chatr +q3p enable ESSSVR`
Solaris, Linux	3.9 GB available by default

Analytic Services uses about 40 MB of memory on startup. In addition, the various caches require the following memory allocations:

- Outline paging cache: 8 MB
- Aggregate storage data cache: 32 MB
- Aggregate storage aggregation cache: 10 MB

Therefore, the initial memory footprint for Analytic Services is about 90 MB. In addition, memory has to be allocated to process incoming query requests. A typical amount of memory to reserve for this purpose is about 300 MB. The total memory allocated for Analytic Services is therefore 390 MB.

On a Windows system with 1.85 GB of addressable memory, the amount available to build and load the outline is about 1.46 GB (1.85 GB - 390 MB = 1.46 GB).

The maximum size of an outline depends on whether the outline is built using a dimension build or is built from an outline already loaded into Analytic Services.

Dimension Build Limit

To build the outline by using a dimension build, Analytic Services allocates about 100 bytes per member, plus the size of the member name, plus the size of all alias names for the member (up to 10 aliases are allowed).

For a sample outline (using a single byte codepage) where the average member name is 15 characters and there is one alias (of 20 characters) per member, the memory requirement for each member that is added is:

100 + 15 + 20 bytes = 135 bytes

The total number of members that can be added in a dimension build is the available memory (1.46 GB, or 153092060 bytes) divided by the number of bytes per member (135), which equals approximately 11 million members.

On systems with more than 2 GB of addressable memory, the outline can be larger in proportion to the extra memory that is available.

When the dimension build is complete, a *databaseName*`.otn` file is saved in the database directory. The `.otn` file is used as input for the outline restructuring process, which replaces the old outline with the new one. During restructuring, two copies of the outline are loaded into memory, the old one (potentially empty), and the new one, so the maximum size of an outline that can be restructured depends on the size of the old outline.

In a dimension build, which starts with an empty outline, only one outline is loaded into memory.

Loaded Outline Limit

The memory requirement for an outline loaded into Analytic Services at runtime or during restructuring is different from the memory requirements for a dimension build. Analytic Services allocates about 60 bytes per member, plus the size of the member name plus 5 bytes, plus the size of all alias names for the member (up to 10 aliases are allowed) plus 5 bytes. For a sample outline where the average member name is 15 characters and there is one alias (of 20 characters) per member, the memory requirement for each member that is added is:

60 + 15 + 5 + 20 + 5 bytes = 105 bytes per member

Assuming 1.46 GB of available memory, the maximum size of an outline that can be loaded is one with 14 million members (1.46 GB / 105 bytes).

The 14 million members are the sum of two outlines that are loaded during restructuring. For example, if an existing outline has 5 million members, the new outline can have a maximum of 9 million members. In an incremental dimension build, it is recommended to build the smaller dimensions first and the larger ones last to allow for a maximum outline size.

Optimizing Outline Paging

Depending on how you want to balance memory usage and data retrieval time, you can customize outline paging for aggregate storage outlines by using one or more of the following settings in the `essbase.cfg` file:

- PRELOADMEMBERNAMESPACE to turn off preloading of the member namespace. See "Member and Alias Namespaces" on page 1304.
- PRELOADALIASNAMESPACE to turn off preloading of the alias namespace. See "Member and Alias Namespaces" on page 1304.
- OPGCACHESIZE to change the size of the outline paging cache. See "Outline Paging Cache" on page 1305.

For specific information on these configuration file settings for outline paging, see the *Technical Reference*.

Member and Alias Namespaces

When Analytic Services loads an outline, it attempts to load into memory the namespaces for both member names and all alias tables to allow optimal performance during name lookup. Name lookup is used during data load, and during report, spreadsheet, and MDX queries.

If the available memory does not allow all namespaces to be preloaded, the alias name space is left on disk and paged in on demand. If there is still not enough memory, the member namespace is also left on disk and paged in on demand.

If memory calculations indicate that it is possible to build and restructure a particular outline if one or both namespaces are not preloaded, you can set one or both of the PRELOADMEMBERNAMESPACE and PRELOADALIASNAMESPACE configuration settings to FALSE to turn off preloading the namespaces.

Not preloading the namespaces will have a severe performance impact but could be a temporary measure to be build a very large outline. After the outline is built and restructured Analytic Services can be restarted with the namespace settings set back to their default TRUE (on) setting.

Outline Paging Cache

For aggregate storage outlines, member data resides in pages in the outline (`.otl`) file. These pages are brought into memory whenever some member information is needed. To maximize performance for outline paging, pages are loaded into the outline paging cache. The outline paging cache uses a 'least recently used' algorithm (LRU), which means pages loaded into the cache to satisfy a member data request remain in the cache until they have to make room for new pages to be brought in.

Page content is organized to provide maximum locality of reference, that is, a good number of a member's siblings are most likely on the same page. Therefore, subsequent requests for the same data or even similar data are executed much more quickly than they would be otherwise.

The default cache size is 8MB and the page size for all supported operating systems is 8192 bytes. Therefore, 1024 pages can be loaded into the cache at any one time.

For very large outlines, performance may be improved by increasing the cache size. To find out if it makes sense to increase the cache you must determine the cache hit rate (the percent of requested pages that are found in the cache).

➤ To determine the cache hit rate, use any of the following methods:

Tool	Topic	Location
MaxL	**query application**	*Technical Reference*
ESSCMD	GETOPGCACHESTATISTICS	*Technical Reference*

For example, use the following MaxL statement to see information on the effectiveness of the current setting for outline paging cache size (including the number of pages read into the cache, the number of pages found in the cache, and the hit rate):

```
query application APP-NAME get opg_cache statistics;.
```

If the hit rate is low, increasing the cache size might make sense. However, increasing the cache size requires additional memory, which impacts the number of members that can be loaded.

Outline Paging File

For aggregate storage outlines, the outline file (`.otl`) is pageable and contains outline data organized in pages of 8192 bytes. Pageable outline files are larger than traditional outline files. Although the data is packed in a way to provide maximum page fill, empty space in the pages results in larger files.

The outline paging file is organized into the following sections, each of which contains a specific type of outline data:

- Member Data
- Member Name Name Space
- Member Formula
- Member UDA (User defined attributes)
- Member UDA Name Space
- Attribute to Base Member Association
- Member Comment
- Extended Member Comment
- Member Alias Name Space

Each dimension in the outline contains all of these sections except for the namespaces. For each alias table there is one member namespace and one alias namespace. The sections are organized as balanced binary trees (BTREE).

You cannot view the content of outline paging files directly. Rather, you must use a MaxL statement or ESSCMD command.

➤ To view information about the outline paging file, use any of the following methods:

Tool	Topic	Location
MaxL	**query database**	*Technical Reference*
ESSCMD	GETOPGDIMSTATISTICS	*Technical Reference*

For example, to display information about the outline paging file, including BTREE information, number of keys, number of pages, and number of levels in the tree, use the following MaxL statement:

```
query database DBS-NAME get opg_state of TABLE-TYPE
   for dimension DIM-NAME
```

Note: The information about the outline paging file is mostly valuable to the Analytic Services technical support team. However, you might find the number of keys to be useful. In any given section, the number of keys is equivalent to the number of records in that section. For example, the number of keys in the Member Formula section is the number of formulas for that dimension. Likewise, the number of keys in the Member Data section is the number of members in that dimension, so you can use query database to see the number of members for each dimension in the outline.

Choosing an Accounts Dimension

For aggregate storage databases, only the accounts dimension (the dimension tagged as accounts) can contain member formulas. All calculations performed on the accounts dimension happen at data retrieval time (when the database is queried). In most cases, the need for formulas on the dimension dictates which dimension you choose to tag as the accounts dimension. However, if there is more than one option for which dimension you tag as accounts, use the following criteria in the specified order:

1. Choose a flat dimension, a dimension that does not have many levels (for example, less than three levels).
2. Of two equally flat dimensions, choose the densest dimension, the dimension with the highest probability that data exists for every combination of dimension members.
3. Of two equally flat, dense dimensions, choose the larger dimension.

If you are converting a block storage outline to an aggregate storage outline and the dimension currently tagged as accounts contains only the addition (+) operator, consider choosing a different dimension as the accounts dimension or having no accounts dimension. Analytic Services uses the accounts dimension to enable data

compression, which is most effective when the accounts dimension is dense. Therefore, an outline with no accounts dimension may be a good choice for outlines that have the following characteristics:

- All dimensions have a low probability that data exists for every combination of dimension members (the more important criteria).
- All dimensions have hierarchies of more than two levels.

Attribute Dimensions

This topic provides information on the differences between aggregate storage databases and block storage databases in regard to attribute dimensions. To use the information in this topic, you should be familiar with attribute dimension concepts for block storage databases. For information about these concepts, see Chapter 10, "Working with Attributes."

The following information applies to attribute dimensions when used on aggregate storage databases:

- Only the Sum calculation is available for attribute calculations.
- Attribute dimensions can be associated with any dimension except the dimension tagged as accounts.
- Attribute dimensions have no impact on data load time or aggregated storage database size.
- Retrievals that query attribute dimensions are calculated at retrieval time.

Note: Calculation order may affect calculation results. Aggregate storage calculation order and block storage calculation order may differ. For information on aggregate storage calculation, see Chapter 59, "Loading, Calculating, and Retrieving Aggregate Storage Data".

Verifying Outlines

Aggregate storage outline files have the same file extension (`.otl`) as block storage database outline files and are stored in an equivalent directory structure. When you save an outline, Analytic Services verifies it for errors. You can also verify the accuracy of an outline before you save it. Some block storage database features do not apply to aggregate storage databases, and the verification process considers the rules for aggregate storage databases. For a full list of outline feature differences, see Chapter 57, "Comparison of Aggregate and Block Storage".

➤ To verify an aggregate storage outline, use the following method:

Tool	Topic	Location
Administration Services	Verifying Outlines	*Essbase Administration Services Online Help*

Developing Formulas on Aggregate Storage Outlines

Formulas calculate relationships between members in a database outline. If you are familiar with using formulas on block storage outlines, keep in mind the following differences when using formulas on aggregate storage outlines:

- Apply formulas directly to members in the database outline. For block storage databases, formulas can be placed in a calculation script. You cannot place formulas in a calculation script.
- Apply formulas to members of the dimension tagged as accounts. In block storage outlines, formulas can be written for members in any dimension.

Analytic Services provides a native calculation language (referred to as the Calc language, or Calc) to write formulas on block storage outlines. To write formulas for aggregate storage outlines, the MDX (Multidimensional Expressions) language is required.

The current chapter concentrates on using MDX to write formulas on aggregate storage databases. For information about using MDX to write queries, see Chapter 35, “Writing MDX Queries.” For information about writing formulas for block storage outlines, see Chapter 22, “Developing Formulas.”

For a reference to MDX functions and syntax, see the MDX section of the *Technical Reference*.

Using MDX Formulas

An MDX formula must always be an MDX numeric value expression. In MDX, a numeric value expression is any combination of functions, operators, and member names that does one of the following:

- Calculates a value
- Tests for a condition
- Performs a mathematical operation

A numeric value expression is different from a set expression. A set expression is used on query axes and describes members and member combinations. A numeric value expression is used to specify a value.

A numeric value expression is used in queries to build calculated members. Calculated members are logical members created for analytical purposes in the WITH section of the query, but which do not exist in the outline.

The following query defines a calculated member and uses a numeric value expression to provide a value for it:

```
WITH MEMBER
 [Measures].[Prod Count]
AS
 'Count (
    Crossjoin (
     {[Units]},
     {[Products].children}
    )
  )'
SELECT
 {[Geography].children}
ON COLUMNS,
 {
  Crossjoin (
     {[Units]},
     {[Products].children}
    ),
   ([Measures].[Prod Count], [Products])
 }
ON ROWS
FROM
 ASOSamp.Sample
```

In the sample query, the WITH clause defines a calculated member, Product Count, in the Measures dimension, as follows:

```
WITH MEMBER
 [Measures].[Prod Count]
```

The numeric value expression follows the WITH clause and is enclosed in single quotation marks. In the sample query, the numeric value expression is specified as follows:

```
'Count (
    Crossjoin (
     {[Units]},
     {[Products].children}
    )
  )'
```

Note: For an explanation of the syntax rules used to build the numeric value expression in the example, see the documentation in the *Technical Reference* for the Count, CrossJoin, and Children functions.

A numeric value expression can also be used as an MDX formula to calculate the value of an existing outline member.

Therefore, rather than creating the example query, you can create an outline member on the Measures dimension called Prod Count that is calculated in the outline in the same way that the hypothetical Prod Count was calculated in the sample query.

➤ To create a calculated member with a formula:

1. Create a member somewhere in the dimension tagged as accounts.
2. Attach an MDX formula to the member.

   Assuming that you created the example Prod Count member, you would use the following formula, which is the equivalent of the numeric value expression used to create the calculated member in the example query:

   ```
   Count(Crossjoin ( {[Units]}, {[Products].children}))
   ```

3. Verify the formula by verifying the outline.

When you retrieve data from the aggregate storage database, the formula is used to calculate the member value.

Before applying formulas to members in the outline, you can write MDX queries that contain calculated members. When you can write an MDX query that returns the calculated member results that you want, you are ready to apply the logic of the numeric value expression to an outline member and validate and test the expression. For information about writing MDX queries, see Chapter 35, "Writing MDX Queries". For syntax information about MDX, see the MDX section of the *Technical Reference*.

Formula Calculation for Aggregate Storage Databases

Formula calculation is much simpler for aggregate storage databases than for block storage databases. Analytic Services calculates formulas in aggregate storage outlines only when data is retrieved.

Calculation order may affect calculation results. For information on calculation order for aggregate storage databases, see "Calculation Order" on page 1335.

Formula Syntax for Aggregate Storage Databases

When you create member formulas for aggregate storage outlines, make sure that you observe the following rules:

- Enclose a member name in braces (`[]`) if the member name meets any of the following conditions:
  - Starts with a number or contains spaces; for example, `[100]`. Braces are recommended for all member names, for clarity and code readability.
  - Is the same as an operator or function name. See the *Technical Reference* for a list of operators and functions.
  - Includes any non-alphanumeric character; for example, a hyphen (`-`), an asterisk (`*`), or a slash (`/`).
- Use the IIF function to write conditional tests with a single else condition. The syntax for the IIF function does not require an ELSEIF keyword to identify the else condition nor an ENDIF keyword to terminate the statement. You can nest IIF functions to create a more complex formula.
- Use the CASE, WHEN, THEN construct to write conditional tests with multiple conditions.
- Be certain that tuples are specified correctly. A tuple is a collection of members with the restriction that no two members can be from the same dimension. Enclose tuples in parentheses; for example, `(Actual, Sales)`.

- Be certain that sets are specified correctly. A set is an ordered collection of one or more tuples. When a set has more than one tuple, the following rule applies: In each tuple of the set, members must represent the same dimensions as do the members of other tuples of the set. Additionally, the dimensions must be represented in the same order. In other words, all tuples of the set must have the same *dimensionality.*

  See "Rules for Specifying Sets" on page 749 for more information about sets.

  Enclose sets in curly braces, for example:

  ```
  { [Year].[Qtr1], [Year].[Qtr2], [Year].[Qtr3],
  [Year].[Qtr4] }
  ```

Creating Formulas on Aggregate Storage Outlines

You use Formula Editor to create formulas. Formula Editor is a tab in the Member Properties dialog box in Outline Editor. Formulas are plain text. You can type the formulas directly into the formula text area, or you can create a formula in the text editor of your choice and paste it into Formula Editor.

You can also include formulas in a dimension build data source. For information, see "Setting Field Type Information" on page 381.

➤ To create a formula, use the following method:

Tool	Topic	Location
Administration Services	Creating Formulas for Aggregate Storage Databases	*Essbase Administration Services Online Help*

Checking Formula Syntax

Analytic Services includes MDX-based syntax checking that tells you about syntax errors in formulas. For example, Analytic Services tells you if you have mistyped a function name or specified a non existent member. Unknown names can be validated against a list of function names. If you are not connected to Analytic Server or to the application associated with the outline, Analytic Services may connect you to validate unknown names.

Syntax checking occurs when you save a formula. Errors are displayed in the Messages panel. If an error occurs, you are given a choice to save or not save the formula. If you save a formula with errors, you are warned when you verify or save the outline. When you calculate a formula with errors, the formula is ignored and the member is given a value of $MISSING.

A syntax checker cannot tell you about semantic errors in a formula. Semantic errors occur when a formula does not work as you expect. One way to find semantic errors in a formula is to place the numeric value expression that defines the formula into a query and run the query to verify that the results are as you expect. See "Using MDX Formulas" on page 1310 to see how to place a formula into a query.

You can use MDX Script Editor to create a query. MDX Script editor provides features such as color coding and auto-completion of MDX syntax. See "About MDX Script Editor" in *Essbase Administration Services Online Help*.

Displaying Formulas

➤ To display a formula, use any of the following methods:

Tool	Topic	Location
Administration Services	Creating Formulas for Aggregate Storage Databases	*Essbase Administration Services Online Help*
ESSCMD	GETMBRCALC	*Technical Reference*
MaxL	**query database**	*Technical Reference*

Composing Formulas on Aggregate Storage Outlines

The following topics discuss and give examples of how to write a variety of formulas for members in aggregate storage outlines:

- "Basic Equations for Aggregate Storage Outlines" on page 1315
- "Members Across Dimensions in Aggregate Storage Outlines" on page 1315

- "Conditional Tests in Formulas for Aggregate Storage Outlines" on page 1316
- "Specifying a User-Defined Attribute in a Formula for Aggregate Storage Outlines" on page 1316

Basic Equations for Aggregate Storage Outlines

You can apply a mathematical operation to a formula to create a basic equation. For example, the following formula is applied to the Avg Units/Transaction member in the ASOsamp Sample database:

```
[Units]/[Transactions]
```

The formula in Avg Units/Transaction divides the number of units by the number of transactions to arrive at the average number of units per transaction.

In aggregate storage outlines, members cannot be tagged as expense items. Therefore, functions in Calc, such as @VAR and @VARPER, which determine the difference between two members by considering expense tags, are not relevant in aggregate storage outlines.

The MDX subtraction operator can be used to calculate the difference between two members. For example, the following formula can be applied to a new member, called Price Diff, in ASOsamp Sample to calculate the difference between the price paid and the original price:

```
[Price Paid]-[Original Price]
```

Members Across Dimensions in Aggregate Storage Outlines

ASOsamp Sample provides a formula on a member called % of Total. This member formula identifies the percentage of the Measures total that is produced by Transactions. The formula for % of Total is as follows:

```
Transactions/(Transactions,Years,Months,
[Transaction Type],[Payment Type],Promotions,Age,
[Income Level],Products,Stores,Geography)
```

The formula specifies a member (Transactions) divided by a tuple (Transactions, Years, ...). The formula lists a top member from every dimension to account for all Transaction data in the cube; that is, not Transaction data for the Curr Year member but Transaction data for all members of the Years dimension, not Transaction data

for months in the first two quarters but Transaction for all months, and so on. In this way, the value of % of Total represents the percentage of the Measures total that are produced by Transactions.

Conditional Tests in Formulas for Aggregate Storage Outlines

You can define a formula that uses a conditional test or a series of conditional tests to determine the value for a member. Use the IIF function to perform a test with a single else condition. You can nest IIF functions to create a more complex query.

The example specifies a formula for a member that represents the price the company must pay for credit card transactions, which includes a 5% charge. The following example assumes that the Credit Price member has been added to the Measures dimension of the ASOsamp Sample database. Credit Price has the following formula, which adds 5% to Price Paid when the payment type is a credit card.

```
IIF (
     [Payment Type].CurrentMember=[Credit Card],
     [Price Paid] * 1.05, [Price Paid]
)
```

Use the CASE, WHEN, THEN construct to create formulas with multiple tests and else conditions.

The Filter function returns the tuples of the input set that meet the criteria of the specified search condition. For example, to establish a baseline (100) for all products you can add a Baseline member and create a formula for it, as follows:

```
Count(Filter(Descendants([Personal
Electronics],[Products].Levels(0)),[Qtr1] > 100.00))
```

Specifying a User-Defined Attribute in a Formula for Aggregate Storage Outlines

User-defined attributes (UDAs) are words or phrases that you create for a member. For example, in Sample Basic, top level members of the Market dimension have the UDA Small Market or the UDA Major Market.

The Major Market example used in this topic shows how to create a formula for a member that shows the sum of sales for all major market members. The example assumes that a new member (Major Market Total) has been added to Sample Basic.

1. MDX provides a Boolean function, IsUDA, which Returns TRUE if a member has the associated UDA tag. The following syntax returns TRUE if the current member of the Market dimension has the UDA "Major Market":

   ```
   IsUda([Market].CurrentMember, "Major Market")
   ```

2. A Filter function, when used with IsUDA (as shown in the following syntax), cycles through each member of the Market dimension and returns a value for each member that has the Major Market UDA:

   ```
   Filter([Market].Members, IsUda([Market].CurrentMember,
   "Major Market"))
   ```

3. The Sum function is used to add the values returned by the Filter function, and for the Major Market example, the following formula is produced:

   ```
   Sum (Filter([Market].Members, IsUda([Market].CurrentMember,
   "Major Market")))
   ```

   This is the formula that is attached to the Major Market Total member.

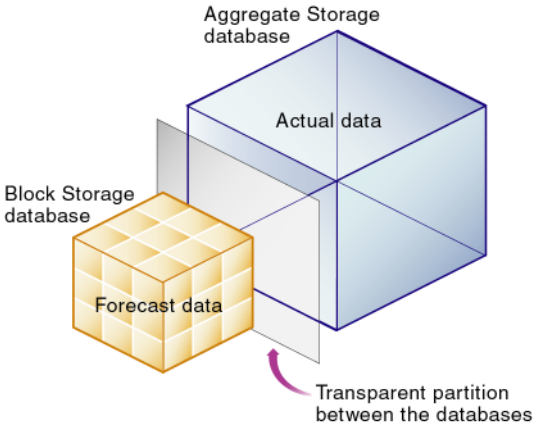
Using a Transparent Partition to Enable Write-Back for Aggregate Storage Databases

This topic discusses creating a transparent partition that connects an aggregate storage database and a block storage database. The partition, called a write-back partition, enables the user to update data on-the-fly (for example, from Spreadsheet Add-in) in the block storage database while data in the aggregate storage database remains unchanged. See Figure 248. Creating a write-back partition potentially decreases calculation time and reduces database size.

To use the information in this topic, you should be familiar with the concepts of partitioned applications. For information about these concepts, see Chapter 13, "Designing Partitioned Applications"

Note: Partitioning is licensed separately from Analytic Services.

Figure 248: Conceptual Diagram Showing a Transparent Partition Used for Analyzing Variance Between Forecast and Actual Data



➤ To create an aggregate storage and block storage write-back partition, use any of the following methods:

Tool	Topic	Location
Administration Services	Aggregate Storage Partition Wizard Creating Partitions	*Essbase Administration Services Online Help*
MaxL	**create database, create partition**	*Technical Reference*

You need Database Designer permissions to create a partitioned application.

Workflow for Creating Write-Back Partitions

This topic provides a high-level workflow for creating write-back partitions. The Administration Services Aggregate Storage Partition Wizard guides you through the following steps:

1. Select or create the aggregate storage database.
2. Create the block storage database in a different application other than the one in which the aggregate storage database is located. Typically the block storage database contains a subset of the dimensions in the aggregate storage database.
3. Create a transparent partition based on where you want the data to be stored. Make the block storage database the target and the aggregate storage database the source.

   Note: You may want to partition on the time dimension if data for some time periods is stored in the aggregate storage database and data for other time periods is stored in the block storage database. For example, if you have actual data for January through March, which is stored in an aggregate storage database, and you want to budget for the last nine months of the year using write-back members in a block storage database.

Users query and write back to the block storage database. Queries are processed by the block storage database or transparently by the aggregate storage database.

For detailed information on using the Administration Services Aggregate Storage Partition wizard to create write-back partitions, see "Aggregate Storage Partition Wizard" in *Essbase Administration Services Online Help*.

Example of Use of a Write-Back Partition

Consider a situation in which you want to compare (do variance analysis) between forecast data and actual data. The following example is based on the ASOsample Sample database, which is provided with Analytic Services. You can recreate the example by using the Administration Services Aggregate Storage Partition wizard to guide you through the process described in this topic.

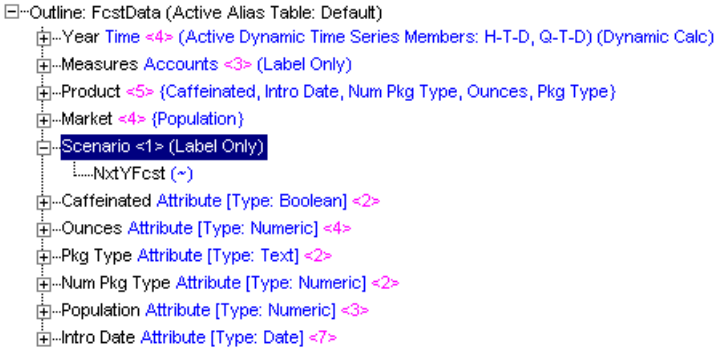
➤ To create a write-back partition for ASOsample sample, complete the following steps:

1. Select the aggregate storage ASOsample Sample database, which contains the actual data for the current year and for previous years; see Figure 249.

   Figure 249: ASOsample Sample Aggregate Storage Database Outline

2. Create a block storage database containing the following subset of dimensions from ASOsample Sample: Measures, Years, Months, Products, Stores, Geography; see Figure 250.
3. Edit the Years dimension to add the following two additional members to the block storage database outline; see Figure 250:
   - A Forecast member to contain the forecast data.
   - A Variance member with a formula to calculate the variance between actual data and forecast data.

Figure 250: Block Storage Database Outline Showing Years Dimension Members for Calculating Variance Between Actual and Forecast Data



4. Delete the Measures dimension member formulas on Avg Units/Transaction and % of Total. These formulas are expressions written in MDX that are copied over from the aggregate storage database. MDX formula expressions cannot be interpreted in block storage databases.
5. Link the databases with a transparent partition on the Years dimension. The block storage database (forecast data) as the target. The aggregate storage database (actual data) as the source. Do not include the write-back members (Forecast and Variance) in the partitioned area.

   Note: If you are using the Administration Services Aggregate Storage Partition wizard this step is done automatically. The databases are automatically partitioned on the Years dimension because you selected the Years dimension in step 3. The write-back members are not included in the partitioned area.

You input forecast values into the block storage database write-back members. Because the added members are outside the partitioned area, you can write to them and then calculate data and generate reports based on updated data. The transparent partition provides a seamless view of both databases.

For detailed information on using the Administration Services Aggregate Storage Partition wizard to create write-back partitions, see “Aggregate Storage Partition Wizard” in *Essbase Administration Services Online Help*.

Chapter

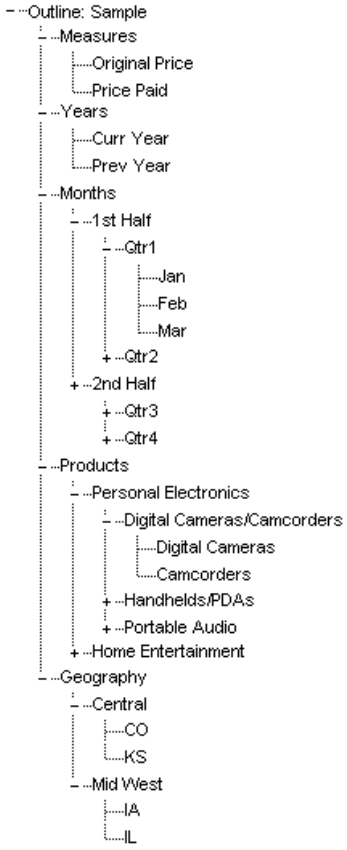
59

Loading, Calculating, and Retrieving Aggregate Storage Data

The most common processes for working with database information include maintaining the outline, loading data values to the database, calculating values, and retrieving database information. In the following topics, this chapter describes how performing these tasks with aggregate storage databases is different from performing these tasks with block storage databases:

Examples in this chapter refer to the outline in Figure 251.

Figure 251: Sample Aggregate Storage Outline



The simplified aggregate storage outline in Figure 251 is not completely expanded. A plus sign (+) node at the left of a member name indicates that the member has children that are not displayed.

Preparing Aggregate Storage Databases

The following topics describe dimension build and data load process differences between aggregate storage databases and block storage databases:

- "Building Dimensions in Aggregate Storage Databases" on page 1325
- "Loading Data into Aggregate Storage Databases" on page 1328
- "Combining Data Loads and Dimension Builds" on page 1333

To use the information in these topics, you should be familiar with data load, dimension build, and rules file concepts and procedures. For information about using data sources to change outlines and to load data values, see Chapter 16, "Understanding Data Loading and Dimension Building."

Building Dimensions in Aggregate Storage Databases

If a dimension build of an aggregate storage database makes a structural change to the outline, all data values are cleared from the database when the dimension build is finished. See Table 87, "Outline Differences Between Aggregate Storage and Block Storage" on page 1289 for details about outline changes that cause restructuring.

Differences between outline characteristics of block storage outlines and aggregate storage outlines affect data sources and rules files. For example, defining a dimension as sparse or dense is not relevant to aggregate storage outlines.

For details, see the following subtopics of this topic:

- "Rules File Differences for Aggregate Storage Dimension Builds" on page 1326
- "Data Source Differences for Aggregate Storage Dimension Builds" on page 1327

Rules File Differences for Aggregate Storage Dimension Builds

Rules files for building aggregate storage outlines must define only outline properties that apply to aggregate storage outlines. For details about aggregate storage outline features and differences, see Table 87, “Outline Differences Between Aggregate Storage and Block Storage” on page 1289.

Before using a rules file that is defined for a block storage outline with an aggregate storage outline, open the rules file in Data Prep Editor, associate the rules file with the aggregate storage outline, and validate the rules file. For instructions for associating outlines, see “Associating an Outline with an Editor” in *Essbase Administration Services Online Help*.

As you edit rules files for aggregate storage databases, some dimension build rules file options that apply only to block storage databases are displayed in Data Prep Editor dialog boxes. Table 94 lists block storage rules file settings that do not apply to aggregate storage outlines. Entries in rules files for these options are ignored when the rules file is processed.

Table 94: Aggregate Storage Dimension Build Rules File Differences

Rules File Location in the Administration Services Interface	Dimension Build Rules File Options That Do Not Apply to Aggregate Storage Databases
Dimension Build Settings dialog box, Global Settings tab	Data configuration options
Dimension Build Settings dialog box, Dimension Build Settings tab	Existing members option: Do not share
Dimension Properties dialog box, Dimension Properties tab	Dimension types option: Country Two-Pass calculation option Data storage options: • Never share • Dynamic Calc and Store • Dynamic Calc All configuration options

Table 94: Aggregate Storage Dimension Build Rules File Differences

Rules File Location in the Administration Services Interface	Dimension Build Rules File Options That Do Not Apply to Aggregate Storage Databases
Dimension Properties dialog box, Accounts Dimension tab	None of the options on this tab apply.
Field Properties dialog box, Dimension Build Properties tab	Field type options: • Currency name • Currency category Currency functionality does not apply to aggregate storage databases.

For details about the dialog boxes, see *Essbase Administration Services Online Help*.

Data Source Differences for Aggregate Storage Dimension Builds

Data sources for modifying aggregate storage outlines should not include field values that apply only to block storage outlines. Table 95 displays property code values that are recognized in dimension build data sources as consolidation properties for members of aggregate storage databases: Any other code is ignored and + (Add) is assumed. For details about specifying member property codes, see "Using the Data Source to Set Member Properties" on page 379.

Table 95: Valid Consolidation Properties for Members of Aggregate Storage Outlines

Code	Description
%	Express as a percentage of the current total in a consolidation (applies only to accounts dimensions)
*	Multiply by the current total in a consolidation (applies only to accounts dimensions)
+	Add to the current total in a consolidation (applies only to accounts dimensions)
-	Subtract from the current total in a consolidation (applies only to accounts dimensions)

Table 95: Valid Consolidation Properties for Members of Aggregate Storage Outlines

Code	Description
/	Divide by the current total in a consolidation (applies only to accounts dimensions)
~	Exclude from the consolidation (applies only to accounts dimensions)
O	Tag as label only

Currency name and currency category field types are not supported for aggregate storage outlines.

In aggregate storage outlines, formulas must be specified in the same format as MDX numeric value expressions. For information about writing formulas for aggregate storage outlines, see "Developing Formulas on Aggregate Storage Outlines" on page 1309.

Loading Data into Aggregate Storage Databases

The differences between loading data into aggregate storage databases and loading data into block storage databases are described in the following topics:

- "Data Source Differences for Aggregate Storage Data Loads" on page 1328
- "Rules File Differences for Aggregate Storage Data Loads" on page 1329
- "Aggregate Storage Data Load Process" on page 1330

For additional information related to aggregate storage data loads, see Table 90, "Data Load Differences Between Aggregate Storage and Block Storage" on page 1294.

Data Source Differences for Aggregate Storage Data Loads

While processing data source records for loading values into aggregate storage databases, Analytic Services processes source data records only for the level 0 dimension intersections where the accounts dimension member does not have a

formula. The following example shows a data source with records for only level 0 intersections. The last field contains data values; the other fields are level 0 members of their respective dimensions.

```
Jan, Curr Year, Digital Cameras, CO, Original Price, 10784
Jan, Prev Year, Camcorders, CO, Original Price, 13573
```

Analytic Services ignores records that specify upper-level members and, at the end of the data load, displays the number of skipped records. For example, the following record would be skipped because member Mid West is a level 1 member:

```
Jan, Curr Year, Digital Cameras, Mid West, Original Price, 121301
```

Since level 0 cells exist only if they contain values, specifying #MISSING or #MI as a value removes the associated cell if it is present in the database. Sorting data sources is unnecessary because Analytic Server reads and sorts all records internally before loading values to the database.

Rules File Differences for Aggregate Storage Data Loads

Rules file specifications for loading values to aggregate storage databases reflect the aggregate storage data load process. Options that apply only to block storage data loads are grayed out in Data Prep Editor when the rules file is associated with an aggregate storage outline.

For block storage data loads, you choose for each data source, through the rules file, whether to overwrite existing values, add values in the data source to existing values, or subtract them from existing values. For aggregate storage data loads using the aggregate storage data load buffer, you make this choice for all data load sources that are gathered into the data load buffer before they are loaded to the database. For information about the data load process, see “Aggregate Storage Data Load Process” on page 1330.

Before using with an aggregate storage outline a rules file that is defined for a block storage outline, open the rules file in Data Prep Editor, associate it with the aggregate storage outline, and validate the rules file. For instructions, see “Associating an Outline with an Editor” in *Essbase Administration Services Online Help*.

Aggregate Storage Data Load Process

For general information about loading data, see Chapter 19, "Performing and Debugging Data Loads or Dimension Builds."

Aggregate storage databases facilitate analysis of very large dimensions containing up to a million or more members. To efficiently support loading data values into such large databases, Analytic Services enables you to combine the processing of multiple data sources through a temporary aggregate storage data load buffer.

When you take advantage of the aggregate storage data load buffer, Analytic Services sorts and works with the values after all data sources have been read. If multiple records are encountered for any specific data cell, the values are accumulated. Analytic Services then stores the accumulated values—replacing, adding to, or subtracting from existing data values in the database.

Note: For data loads using the aggregate storage data load buffer, the choice for replacing, adding, or subtracting values is specified for the entire set of data sources. For all other data loads, the choice for replacing, adding, or subtracting values is specified per data source through rules files.

Taking advantage of the aggregate storage data load buffer can significantly improve overall data load performance because it requires fewer input/output operations.

The following topics provide an overview of the data load processes based on the method used.

- "Using MaxL to Perform Aggregate Storage Data Loads" on page 1330
- "Using Essbase Administration Services Console to Perform Aggregate Storage Data Loads" on page 1332

Using MaxL to Perform Aggregate Storage Data Loads

Using the MaxL **import database data** statement to load data values from a single data source does not involve the aggregate storage data load buffer. If you use multiple statements in incremental data loads to aggregate storage databases, you can significantly improve performance if you load values to the aggregate storage data load buffer first, with a final write to storage after all data sources have been read.

To use this buffer in MaxL, you specify a series of separate statements that perform the following tasks:

- Prepare the aggregate storage data load buffer to accumulate the data values
- Define and provide the data sources and rules files
- Conclude the operation by writing the accumulated values to the database

A sample of the sequence of statements loading three data sources (file_1, file_2, and file_3) follows:

1. First, an aggregate storage data load buffer is initialized, as defined by the following MaxL **alter database** statement:

   ```
   alter database agg.sample
      initialize load buffer with buffer_id 1;
   ```

2. Next, data sources are read into the aggregate storage data load buffer where they are sorted and accumulated. You can include any combination of data sources, such as `.xls` files, text files, and SQL relational sources. The following three statements show an example of multiple data sources to be accumulated in the buffer:

   ```
   import database agg.sample data
      from server data_file 'file_1.txt'
      to load_buffer with buffer_id 1;
   import database agg.sample data
      from server data_file 'file_2'
      using server rules_file ‘rule’
      to load_buffer with buffer_id 1;
   import database agg.sample data
      from server excel data_file 'file_3.xls'
      to load_buffer with buffer_id 1;
   ```

   Note: Import statements need not be contiguous. As long as the buffer exists, the database is locked from queries, aggregations, or data loads from other means, such as from Administration Services. You can perform other MaxL operations, such as displaying security information or creating triggers.

3. A final import statement loads the buffer contents to the database cells, replacing existing values:

```
import database agg.sample data
   from load_buffer with buffer_id 1
   override values;
```

Note: Performing an application restart loses the data load buffer. When you restart the application, you restart the entire process.

Using Essbase Administration Services Console to Perform Aggregate Storage Data Loads

Essbase Administration Services Console always uses the aggregate storage data load buffer for aggregate storage data loads. In this process, when you initiate a data load you provide all data source file names and rules file names in a single dialog box.

For further information about the methods for loading values to aggregate storage databases, see the following topics:

Tool	Topic	Location
Administration Services	Performing a Data Load or Dimension Build for Aggregate Storage Databases	*Essbase Administration Services Online Help*
ESSCMD	BUILDDIM IMPORT LOADDB	*Technical Reference*
MaxL	**alter database** **import data**	*Technical Reference*

Note: If values have been calculated and stored through an aggregation, Analytic Services automatically updates higher-level stored values when data values are changed. No additional calculation step is necessary. The existence and size of an aggregation can affect the time it takes to perform a data load. For additional information about aggregations, see “Aggregations” on page 1337.

Combining Data Loads and Dimension Builds

When using the aggregate storage data load buffer, you can combine data sources and rules files to add members to the outline and to load data values to the level 0 cells. Regardless of the order you specify the files, Analytic Services first makes the outline changes and then loads the data values.

Calculating Aggregate Storage Databases

Aggregate storage database values are calculated through the outline structure and MDX formulas. When a data load is complete, all the information needed to calculate an aggregate storage database is available. The values to support a retrieval request are calculated by a consolidation of the values loaded for level 0 members. Formulas for accounts dimension members are calculated for each retrieval. Values calculated for retrievals are not stored.

To improve retrieval performance, Analytic Services can aggregate values and store them ahead of time. However, aggregating and storing all values can be a lengthy process that requires disk space for storage. Analytic Services provides an intelligent aggregation process that balances time and storage resources. For details, see “Aggregating an Aggregate Storage Database” on page 1335.

To prepare an aggregate storage database for retrieval, you create the outline and load the level 0 values. Then you calculate the database by aggregating, and storing additional values, with the remaining values to be calculated when retrieved.

Note: If a database needs calculation scripts for special calculations and data dependencies, make it a block storage database.

The following topics further describe calculation of aggregate storage databases:

For additional information related to aggregate storage database calculations, see Table 88, “Calculation Differences Between Aggregate Storage and Block Storage” on page 1293.

Outline Factors Affecting Data Values

The hierarchical structure of an aggregate storage outline determines how values are rolled up. L0 member values roll up to L1 member values, L1 member values roll up to L2 member values, and so on.

Consolidation operators assigned to members of the dimension tagged as accounts define the operations used in the roll-up: add (+), subtract (-), multiply (*), divide (/), percent (%), and no operation (~). For an explanation of operators, see Table 10 on page 161.

Note: In aggregate storage outlines, consolidation operators assigned to members of non-accounts dimensions have no effect on calculation. For non-accounts dimensions, the only allowable consolidation operator is the add (+) operator.

For more complex operations, you can provide MDX formulas for members of the accounts dimension. MDX formulas are written in the same format as MDX numeric value expressions. For information about writing formulas for aggregate storage outlines, see "Developing Formulas on Aggregate Storage Outlines" on page 1309.

Block Storage Calculation Features That Do Not Apply to Aggregate Storage Databases

The following characteristics of calculating block storage databases do not apply to aggregate storage databases:

- Calculation script calculations and concern for monitoring and tracing calculations
- Dynamic Calc and Dynamic Calc and Store member storage properties
- Block storage formula syntax and pre-defined Essbase functions in formulas
- Custom-defined calculation functions and custom-defined calculation macros
- Formulas on members of dimensions other than the accounts dimension
- Preloaded values for member intersections above level 0
- Calculation order and two-pass calculations and concern for dimension order in the outline
- Time balance data values; for example, carrying over the value from the end of a time period, such as Jan, to the beginning value for the next time period, such as Feb

- Time series data values; for example, calculating period-to-date, first, last, and average sales values
- Block storage performance features such as Intelligent Calculation

Calculation Order

Calculation order may affect calculation results. Aggregate storage calculation order and block storage calculation order may differ. For block storage databases, by default, Analytic Services calculates dynamically-calculated members on standard dimensions and then aggregates members along the attribute dimensions.

For aggregate storage databases, Analytic Services calculates data in the following order:

1. Aggregates members on all dimensions except the dimension tagged as accounts.The order in which members and dimensions are aggregated is optimized internally and changes according to the nature of the database outline.
2. Calculates accounts dimension members and formulas.

Because the internal calculation order for an aggregate storage database is not predictable, any inherent rounding errors are also not predictable. These rounding errors are expected behavior in computer calculation and are extremely small in relation to the data values concerned.

Aggregating an Aggregate Storage Database

For aggregate storage databases, after data values are loaded into the level 0 cells of an outline, the database requires no separate calculation step. From any point in the database, users can retrieve and view values that are aggregated for only the current retrieval. Aggregate storage databases are smaller than block storage databases, enabling quick retrieval of data values.

For even faster retrieval, you can precalculate data values and store the precalculated results in aggregations. The following terms are integral to an explanation of aggregate storage database calculation:

- “Aggregate Cells” on page 1336
- “Aggregate Views” on page 1336
- “Aggregations” on page 1337

Aggregate Cells

Cells for level 0 intersections across dimensions, without formulas, are called input cells. Data values can be loaded to them. Higher-level cells of the accounts dimension are always calculated at retrieval. All other higher-level intersections across dimensions are *aggregate cells*. For example, for the outline in Figure 251 on page 1324, Price Paid > Curr Year > 1st Half > Portable Audio > CO is an aggregate cell; Original Price > Curr Year > Jan > Camcorders > CO is another aggregate cell. Values for aggregate cells must be rolled up from lower-level values.

Aggregate cell values are calculated for each request, or they can be precalculated and stored in aggregations.

Aggregate Views

When Analytic Services defines which aggregate cells to precalculate and store, it works with the cells in sets, known as aggregate views. An *aggregate view* is a collection of aggregate cells. The collection is based on the levels of the members within each dimension.

For example, consider one aggregate view for the outline in Figure 251 on page 1324. This aggregate view includes aggregate cells for level 0 members of the Measures, Years, and Geography dimensions, level 1 members of the Months dimension, and level 2 members of the Products dimension. The dimensions are arranged in the outline as Measures, Years, Months, Products, Geography. The example aggregate view is shown as 0,0,1,2,0.

The 0,0,1,2,0 aggregate view contains aggregate cells that include the following member intersections:

```
Original Price, Curr Year, Qtr1, Personal Electronics, CO
Original Price, Curr Year, Qtr1, Personal Electronics, KS
Original Price, Curr Year, Qtr1, Home Entertainment,   CO
Original Price, Curr Year, Qtr1, Home Entertainment,   KS
Original Price, Curr Year, Qtr2, Personal Electronics, CO
Original Price, Curr Year, Qtr2, Personal Electronics, KS
Original Price, Curr Year, Qtr2, Home Entertainment,   CO
Original Price, Curr Year, Qtr2, Home Entertainment,   KS
Original Price, Curr Year, Qtr3, Personal Electronics, CO
Original Price, Curr Year, Qtr3, Personal Electronics, KS
Original Price, Curr Year, Qtr3, Home Entertainment,   CO
Original Price, Curr Year, Qtr3, Home Entertainment,   KS
Original Price, Curr Year, Qtr4, Personal Electronics, CO
Original Price, Curr Year, Qtr4, Personal Electronics, KS
```

```
Original Price, Curr Year, Qtr4, Home Entertainment,   CO
Original Price, Curr Year, Qtr4, Home Entertainment,   KS
Original Price, Prev Year, Qtr1, Personal Electronics, CO
Original Price, Prev Year, Qtr1, Personal Electronics, KS
Original Price, Prev Year, Qtr1, Home Entertainment,   CO
Original Price, Prev Year, Qtr1, Home Entertainment,   KS
and so on...
```

Aggregations

Aggregations are consolidations, based on outline hierarchy, of level 0 data values. An aggregation contains one or more aggregate views. Analytic Services provides an intelligent aggregation process that selects aggregate views to be rolled up, aggregates them, and then stores the values for the cells in the selected views. If an aggregation includes aggregate cells dependent on level 0 values that are changed through a data load, the higher-level values are automatically updated at the end of the data load process.

The term *aggregation* is used for both the process and the result of the process.

Performing Database Aggregations

You can use either Administration Services Aggregation Design Wizard or MaxL statements to perform an aggregation. Internally, the aggregation process has two phases:

- Aggregate view selection.
- Calculation and storage of values for the selected aggregate views. This phase is also called the materialization of the aggregation.

During the aggregate view selection phase, Analytic Services analyzes how calculating and storing various combinations of aggregate views would affect average query response time. Based on their usefulness, Analytic Services creates an internal list of the aggregate views and determines a stopping point in physical storage used that balances performance requirements with the impact on physical storage. If desired, you can specify the amount of physical storage for an aggregation.

The first time that you perform an aggregation after an outline change, it is recommended that you let Analytic Services select the physical storage stopping point. After the calculation and storage phase, you can test retrieval times based on your retrieval requirements. To optimize performance for your situation, you can

rerun the aggregation wherein you can increase or decrease the storage stopping point to be used by the aggregation. Specifying a storage limit reduces or increases the storage space required, the time required to materialize the aggregation, and the time required for some retrievals.

Administration Services Aggregation Design Wizard enables you to create aggregation scripts. An aggregation script represents a specific scenario that is relevant to the existing outline and to aggregate view selection determined by the storage limit, if one is specified. Aggregation Design Wizard displays the selected aggregate views and enables you to save aggregation scripts for later materialization. You can also execute the aggregation script immediately to materialize the aggregate views that it specifies. For additional information about aggregation scripts, see "Working with Aggregation Scripts" on page 1338.

Aggregation Design Wizard also enables running aggregation processes in the background. When you run an Administration Services process in the background you can continue to use Essbase Administration Services Console for other activities at the same time as the background process is running.

MaxL addresses the aggregation selection and materialization phases in a single statement. To specify that Analytic Services determine the stopping point, omit the optional part of the MaxL **execute aggregate process** statement beginning with the keywords **stopping when total_size exceeds**.

➤ To perform a database aggregation, use any of the following methods:

Tool	Topic	Location
Administration Services	Improving Retrievals by Precalculating Aggregations	*Essbase Administration Services Online Help*
MaxL	**execute aggregate process**	*Technical Reference*

Working with Aggregation Scripts

Saved aggregation scripts enable you to materialize an aggregation at a different time than the aggregate views for the aggregation are selected. Support for multiple aggregation scripts also enables you to choose from different selection scenarios. For example, one aggregation script that builds a 40 MB aggregation may support normal retrieval requirements. For a large, detailed weekly report, you could build a 60 MB aggregation that reduces the time to produce the report, then go back to the 40 MB script.

Only Administration Services Aggregation Design Wizard enables you to save aggregation scripts. Aggregation scripts are stored in the database directory as text files with the `.csc` extension and are valid until the outline is changed. To avoid the potential clutter of invalid aggregation script files, when you change the outline, be sure to delete existing aggregation scripts manually or through the Aggregation Design Wizard. You can also rename existing aggregation scripts.

Note: When Aggregation Design Wizard displays the list of existing aggregation scripts, it lists all files with the `.csc` extension in the database directory. Only valid aggregation script files can be executed.

You can use the Aggregation Design Wizard to execute an aggregation script or you can manually copy the contents of the aggregation script to a MaxL script which can be executed at a later time. Executing an aggregation script materializes the aggregate views specified within it. Although you can create multiple aggregation scripts, only one aggregation can be materialized at a time.

Retrieving Aggregate Storage Data

This topic includes the following subtopics:

For additional information related to query support, see Table 91, "Query Differences Between Aggregate Storage and Block Storage" on page 1294.

Attribute Calculation Retrievals

Aggregate storage applications support only the Sum member of the Attribute Calculations dimension. If you specify any other member name from the Attribute Calculations dimension, such as Min or Avg, an error is returned. For information about the Attribute Calculations dimension, see "Understanding the Attribute Calculations Dimension" on page 201.

Retrieval Tools Supporting Aggregate Storage Databases

Analytic Services provides the following programs and tools that enable data retrieval from aggregate storage databases:

- MDX Queries
- Report Writer, which is run through Essbase Administration Services Console or the MaxL export data using report_file grammar
- Essbase Spreadsheet Services
- Essbase Spreadsheet Add-in

Any commands that support features that apply only to block storage cannot be used with aggregate storage databases; for example, the Report Writer <SPARSE command.

Chapter 60

Managing Aggregate Storage Applications and Databases

Most processes for managing aggregate storage applications are the same as for managing block storage applications. This chapter describes differences in the following topics:

See also Table 86 on page 1288 for basic information about how aggregate storage applications are structured and managed.

Aggregate Storage Security

Defining and executing aggregations requires Calculation (Essbase Administration Services) or Execute (MaxL) permission or higher. Dimension builds clear the database and can be performed only by users with Write permissions. In all other areas, security for aggregate storage applications and security for block storage applications is the same.

For information about permissions, see the following documentation:

Topic	Location
"Understanding Security and Permissions" on page 836	*Essbase Analytic Services Database Administrator's Guide*

Topic	Location
"Managing User/Group Permissions for Applications and Databases"	*Essbase Administration Services Online Help*
"Privileges and Roles"	*Technical Reference* From the index, use the keyword privileges, defined.

Backing Up Aggregate Storage Applications

The file structure used by aggregate storage applications to store application and database information differs from the file structure used by block storage applications use. Table 96 lists the major directories and files associated with aggregate storage applications.

Table 96: Aggregate Storage Files

Directory or File	Location	Description
appname	*arborpath*\app\	Application directory
appname.app	*arborpath*\app*appname*\	Application file containing application settings
appname.LOG	*arborpath*\app*appname*\	Application log file
dbname	*arborpath*\app*appname*\	Database directory
dbname.db	*arborpath*\app*appname**dbname*\	Database file containing database settings
dbname.dbb	*arborpath*\app*appname**dbname*\	Backup of database file
dbname.ddb	*arborpath*\app*appname**dbname*\	Partition definition file
dbname.otl	*arborpath*\app*appname**dbname*\	Outline file
trigger.trg	*arborpath*\app*appname**dbname*\	Trigger file
default	*arborpath*\app*appname*\	Tablespace directory: default
temp	*arborpath*\app*appname*\	Tablespace directory: temp

Table 96: Aggregate Storage Files

Directory or File	Location	Description
`log`	`\arborpath\app\appname\`	Log directory
`metadata`	`\arborpath\app\appname\`	Metadata directory

`essn.dat`	`\arborpath\app\appname\default\`
	`\arborpath\app\appname\log\`
	`\arborpath\app\appname\metadata\`
		Data file

➤ To back up an aggregate storage database, ensure that the application is stopped and use the operating system to copy the entire application directory: *arborpath*\app*appname*\.

Note: The ESSCMD command BEGINARCHIVE and the MaxL statement **alter database begin archive** do not support aggregate storage databases.

Managing Storage for Aggregate Storage Applications

For aggregate storage applications, Tablespace Manager controls all aspects of retrieving and storing data. For information about how block storage is managed, see "Understanding the Analytic Server Kernel" on page 1022.

Tablespace manager works with tablespace definitions to manage the physical disk for data storage and work areas. The following topics describe tablespaces and how to work with them:

- "Working with Tablespaces" on page 1344
- "Defining Tablespaces" on page 1345

Working with Tablespaces

Essbase Analytic Services uses tablespaces to optimize data storage and to optimize retrieval for data files and work files. *Tablespaces* define one or more location definitions that map data objects, such as aggregate views and aggregations, to files. Within each application directory, Analytic Services sets up directories for four tablespaces.

You cannot change the location or size of metadata and log, two of the tablespaces. You can specify sizes and multiple locations for the other tablespaces:

- The default tablespace, which contains the database data structure and the database values. After data is loaded to this tablespace, the location cannot be changed.
- The temp tablespace, which provides a temporary work space for Analytic Services to use during various operations, such as data loads, aggregations, and retrievals.

You can define the following tablespace properties:

- Directory path locations
- Maximum disk space to be used at each location
- Maximum file size allowed within each location

Note: You can modify or delete the file locations used to store information within each tablespace. You cannot delete a file location if it contains data.

Defining a maximum disk space for a tablespace location defines an end point for the location. Files cannot be written past the end point. The maximum disk space definition does not reserve the entire amount of disk space specified. As needed, Tablespace Manager allocates disk space in fixed-size increments.

When Analytic Services extends a file or adds a file to a tablespace, it looks for space in the first location. If the disk space is not available, Analytic Services checks the next location, and so on. Analytic Services starts writing files to the first available space within the defined locations. When no space is available, because all file locations are used, an error is returned.

When values in a database are cleared, the files in the tablespaces shrink, releasing disk space. When work files are no longer needed, they are deleted and disk space in the tablespace is released. If space is unused, other computer programs can use it.

Based on the maximum size specified for files, Analytic Services writes multiple files; for example, `ess00001.dat`, `ess00002.dat`, and so on. If you back up database files to other media, you should not set the maximum file size for a tablespace to be greater than the maximum file size that the media can handle.

Defining Tablespaces

You define separate tablespace definitions for each aggregate storage application.

For further information about the methods for defining tablespaces, see the following topics:

Tool	Topic	Location
Administration Services	Managing Tablespaces	*Essbase Administration Services Online Help*
MaxL	**alter tablespace**	*Technical Reference*

Managing the Aggregate Storage Cache

Analytic Services uses the aggregate storage cache to facilitate use of memory during data loads, aggregations, and retrievals. The aggregate storage cache is the only cache relevant to aggregate storage databases.

Note: The cache memory locking feature is used only with block storage applications.

When an aggregate storage outline is started, Analytic Services allocates a small area in memory as the aggregate storage cache for the relevant application. As additional cache area is needed, Analytic Services increases the cache size incrementally until the maximum cache size specified for the application is used or until the operating system denies additional allocations.

Note: After additional aggregate cache memory allocations are denied, use of existing memory can still increase.

You can view the current aggregate cache memory allocation and the maximum aggregate cache size setting. In some situations, changing the current setting can optimize use of memory. By default, the maximum cache size is set at 32 MB, the minimum size for the setting. As a general guideline, you can use the size of input-level data to determine when to increase the maximum size for the cache.

The size of input-level data is the size of level 0 values. The size of input-level data is a database statistic that both Administration Services and MaxL display as an aggregate storage database property.

A 32 MB cache setting supports a database with approximately 2 GB of input-level data. If the input-level data size is greater than 2 GB by some factor, the aggregate storage cache can be increased by the square root of that factor. For example, if the input-level data size is 3 GB (which is 2 GB * 1.5), multiply the aggregate storage cache size of 32 MB by 1.22 (which is approximately the square root of 1.5), and set the aggregate cache size to the result: 39.04 MB.

Another factor to consider is the number of threads set for parallel calculation. Essbase uses multiple threads during the aggregation materialization process. The threads divide up aggregate storage cache. If you increase the number of threads specified in the CALCPARALLEL configuration setting for aggregate storage applications or databases, consider the possible need to increase the size of the aggregate storage cache. For information about the CALCPARALLEL configuration setting, see the *Technical Reference*.

Note: It is possible, for aggregate storage applications, to improve performance by setting the number of threads higher than the number of processors.

Do not increase the maximum size of the aggregate storage cache if a greater size is not needed.

For further information about viewing and setting the aggregate storage cache size for an application, see the following topics:

Tool	Topic	Location
Administration Services	Sizing the Aggregate Storage Cache	*Essbase Administration Services Online Help*
MaxL	**query application** **alter application**	*Technical Reference*

Note: A changed aggregate storage cache setting does not take effect until the application is restarted.

Appendix A

Limits

This appendix describes in the following categories size and quantity limits that you may encounter while working with Analytic Services:

Note: In the following tables, the term non-Unicode applications refers to non-Unicode-mode applications and applications on Analytic Servers that are not Unicode-enabled. For information about Unicode see Chapter 39, "Analytic Services Implementation of Unicode."

Names and Related Objects

Object	Limits
Alias name	• Non-Unicode application limit: 80 bytes • Unicode-mode application limit: 80 characters
Alias table name	• Non-Unicode application limit: 30 bytes • Unicode-mode application limit: 30 characters
Application name	• Non-Unicode application limit: 8 bytes • Unicode-mode application limit: 30 characters
Application description	• Non-Unicode application limit: 79 bytes • Unicode-mode application limit: 80 characters

Object	Limits
• Custom-defined function name • Custom-defined macro name • Custom-defined function specification • Custom-defined macro specification	• Non-Unicode application limit: 127 bytes. MaxL and the API truncate characters after 127 bytes. • Unicode-mode application limit: 128 characters. MaxL and the API truncate characters after 128 characters. In either case, no truncation on server. No error is displayed if truncation occurs.
Custom-defined function and macro comment	• Non-Unicode application limit: 255 bytes. After 255 bytes, characters are truncated by MaxL and API. • Unicode-mode application limit: 256 characters. MaxL and the API truncate characters after 256 characters. In either case, no truncation on server. No error is displayed if truncation occurs.
Database name	• Non-Unicode application limit: 8 bytes • Unicode-mode application limit: 30 characters
Database description	• Non-Unicode application limit: 79 bytes • Unicode-mode application limit: 80 characters
Directory path For example: `/essbase/bin`	• Non-Unicode application limit: 256 bytes • Unicode-mode application limit: 1024 bytes
File name for calculation scripts, report scripts, or rules files	• Non-Unicode application limit: 8 bytes • Unicode-mode application limit: If included within a path, the smaller of the following two values: – 1024 bytes – The limit established by the operating system If not included within a path, as in some MaxL statements, 1024 bytes.
Filter name	• Non-Unicode application limit: 30 bytes • Unicode-mode application limit: 30 characters
Group name	• Non-Unicode application limit: 30 bytes • Unicode-mode application limit: 30 characters
Linked reporting object cell note	• Non-Unicode application limit: 600 bytes • Unicode-mode application limit: 600 characters

Object	Limits
Linked reporting object URL	512 characters (always single-byte characters)
Linked reporting object description for URLs and files	• Non-Unicode application limit: 80 bytes • Unicode-mode application limit: 80 characters
Member comment field	• Non-Unicode application limit: 255 bytes • Unicode-mode application limit: 256 characters
Member name	• Non-Unicode application limit: 80 bytes • Unicode-mode application limit: 80 characters
Analytic Server name	• Non-Unicode application limit: 29 bytes • Unicode-mode application limit: 50 characters
Password	• Non-Unicode application limit: 100 bytes • Unicode-mode application limit: 100 characters
Substitution variable name	320 bytes
Substitution variable value	256 bytes
Trigger name	• Non-Unicode application limit: 30 bytes • Unicode-mode application limit: 30 characters
UDA (user-defined attribute)	• Non-Unicode application limit: 80 bytes • Unicode-mode application limit: 80 characters
User name	• Non-Unicode application limit: 30 bytes • Unicode-mode application limit: 30 characters

Data Load and Dimension Build Limits

Object	Limits
Selection and rejection criteria	Number of characters that describe selection and rejection criteria: combination of all criteria limited to 32 KB
Number of error messages written to a data load or dimension build error log (DATAERRORLIMIT in `essbase.cfg`)	Default 1000, minimum 1, maximum 65000

Other Size or Quantity Limits

Object	Limits
Caches: data, data file, index	2 GB
Calculation script size	Calculation script text including blanks, tabs, and so on: 64 KB
Formula size	• Created in Formula Editor: 64 KB. • Created in MaxL, using multi-byte characters: 40 KB.
Number of aggregate views within an aggregation	1023
Number of security filters	• Per Analytic Server, 65535 • Per Analytic Services database, 32290
Number of users	30,000. Errors can occur if you create more than 30,000 users.
Number of members in a block storage outline	Approximately 1,000,000 explicitly defined in an Analytic Services outline. Hybrid Analysis and some uses of partitions enable access to many more members than are explicitly listed in an outline, the actual number of members accessible through the database is much higher. Longer names, such as can occur if multi-byte characters are used, decrease the number of members that are allowed.
Number of members in an aggregate storage outline	10,000,000 to 20,000,000 members, depending on available memory and other memory requirements.
Number of dimension levels in an aggregate storage outline	The product of the numbers of dimension levels across all dimensions cannot exceed 2^{32} which is equal to 4,294,967,296. For example, to calculate this number for a nine dimension outline with dimension levels 5,10,20,20,20,5,5,5, multiply the number of levels of the first dimension by the number of levels of the second dimension by the number of levels of the third dimension, and so on. 5*10*20*20*20*5*5*5*5 = 1,250,000,000, which is less than 4,294,967,296.

Appendix

B

Handling Errors and Troubleshooting Analytic Services

This appendix describes tools that you can use to diagnose errors, and suggests methods for correcting some errors:

Note: Chapters related to partitions, currency conversion, and data load have basic troubleshooting information in the relevant chapters. For information about restoring from backups, see Chapter 47, "Backing Up and Restoring Data." For information about specific error messages, see the "Error Messages Guide" in `/docs/errmsgs` in the Essbase installation. For information about error and exception logs, see Chapter 43, "Monitoring Data, Applications, and Databases."

Understanding Fatal Error Handling

The Analytic Server Kernel considers the following errors fatal:

- One or more control fields have unexpected or inconsistent values.
- The kernel detects data corruption.
- The kernel cannot perform an operation that is necessary to ensure data integrity (for example, disk space is insufficient).
- The kernel encounters a condition that can lead to data corruption.

When the kernel encounters a fatal error, it shuts down and restarts, attempting to reinitialize itself and proceed with database recovery. When recovery begins, Analytic Services displays an error message similar to this one:

```
1080022 Reinitializing the Essbase Kernel for database
database_name due to a fatal error...
```

This message is followed by other informational messages related to database recovery, such as this one:

```
1080028 Performing transaction recovery for database
database_name during fatal error processing.
```

When you see such messages, you know that the kernel shut itself down and is attempting to start up again. Check the Analytic Server log and determine whether Analytic Services issued a fatal error message just before it generated the reinitialization messages. For a review of methods used to view the Analytic Server log, see "Using Analytic Server and Application Logs" on page 993.

If the kernel did encounter a fatal error, in most cases you need to restart any operation that was active at the time of the fatal error. If the operation was a calculation or a data load, you may be able to continue where the operation left off; check the Analytic Server log to see how far Analytic Services processed the operation. When in doubt, restart the operation. For descriptions of types of server interruptions and of recovery procedures, see "What to Expect If a Server Interruption Occurs" on page 1071.

If the kernel did not encounter a fatal error, contact your software provider's technical support to determine what caused the kernel to shut down and restart.

See "Contents of the Analytic Server Log" on page 979 for detailed descriptions of the contents of the Analytic Server log. See "Analytic Server and Application Log Message Categories" on page 991 for specific information about identifying the component where the error occurred.

Recovering from Full Restructure Failure

If an error or system failure occurs while Analytic Services is restructuring, it is most likely to occur during step 2 in the procedure "Full Restructures" on page 1150.

➤ To recover from a failure during step 2 of "Full Restructures" on page 1150:

1. Delete the temporary files, both to free up disk space and to avoid conflicts the next time you restructure the database.
2. Restart the database.

➤ To recover from a failure during step 1, step 3, or step 4 of "Full Restructures" on page 1150:

1. Review the disk directory and determine how far the restructuring has progressed.
2. If all but step 4 is complete, rename the temporary files to the correct file names.

Recovering from Sparse Restructure Failure

If a system failure occurs during any step of a sparse restructure, you can recover by restarting the database.

Synchronizing Member Names in Report Scripts and Database Outlines

When you run a report, it is important to ensure that the member names in the report script match the member names in the database outline. An error displays every time the Report Extractor cannot find a matching member name, and you must correct the name in the report script before the report continues processing.

Handling Analytic Server Problems When Running Multiple Reports

Your server machine may freeze if you try to run more reports in parallel than you have assigned server thread resources.

If you are running multiple report scripts and your server freezes, check the value of the configuration file setting SERVERTHREADS in the `essbase.cfg` file on the server. There should be at least one thread for each report running. For example, if you are running 22 reports, the value for SERVERTHREADS should be at least 22.

Appendix C

Estimating Disk and Memory Requirements

This appendix helps you estimate disk and memory requirements. This appendix contains the following sections:

This appendix uses a worksheet approach to help you keep track of the many components that you calculate. If you are using the printed version of this book, you can photocopy the worksheets. Otherwise, you can simulate the worksheets on your own paper. Labels, such as DA and MA help you keep track of the various calculated disk and memory component values.

Note: The calculations in this appendix apply only to block storage databases.

Understanding How Analytic Services Stores Data

You need to understand the units of storage that Analytic Services uses in order to size a database. This discussion assumes that you are familiar with the following basic concepts before you continue:

- How Analytic Services stores data, as described in Chapter 4, "Basic Architectural Elements"
- How Analytic Services Kernel components manage Analytic Services data, as described in Chapter 44, "Managing Database Settings"

An Analytic Services database consists of many different components. In addition to an outline file and a data file, Analytic Services uses several types of files and memory structures to manage data storage, calculation, and retrieval operations.

Table 97 describes the major components that you must consider when you estimate the disk and memory requirements of a database. “Yes” means the type of storage indicated is relevant, “No” means the type of storage is not relevant.

Table 97: Storage Units Relevant to Calculation of Disk and Memory Requirements

Storage Unit	Description	Disk	Memory
Outline	A structure that defines all elements of a database. The number of members in an outline determines the size of the outline.	Yes	Yes
Data files	Files in which Analytic Services stores data values in data blocks in data files. Named `ess`*`xxxxx`*`.pag`, where *xxxxx* is a number. Analytic Services increments the number, starting with `ess00001.pag`, on each disk volume. Memory is also affected because Analytic Services copies the files into memory.	Yes	Yes
Data blocks	Subdivisions of a data file. Each block is a multidimensional array that represents all cells of all dense dimensions relative to a particular intersection of sparse dimensions.	Yes	Yes
Index files	Files that Analytic Services uses to retrieve data blocks from data files. Named `ess`*`xxxxx`*`.ind`, where *xxxxx* is a number. Analytic Services increments the number, starting with `ess00001.ind`, on each disk volume	Yes	Yes
Index pages	Subdivisions of an index file. Contain index entries that point to data blocks. The size of index pages is fixed at 8 KB.	Yes	Yes
Index cache	A buffer in memory that holds index pages. Analytic Services allocates memory to the index cache at startup of the database.	No	Yes

Table 97: Storage Units Relevant to Calculation of Disk and Memory Requirements (Continued)

Storage Unit	Description	Disk	Memory
Data file cache	A buffer in memory that holds data files. When direct I/O is used, Analytic Services allocates memory to the data file cache during data load, calculation, and retrieval operations, as needed. Not used with buffered I/O.	No	Yes
Data cache	A buffer in memory that holds data blocks. Analytic Services allocates memory to the data cache during data load, calculation, and retrieval operations, as needed.	No	Yes
Calculator cache	A buffer in memory that Analytic Services uses to create and track data blocks during calculation operations.	No	Yes

Determining Disk Space Requirements

Analytic Services uses disk space for its server software and for each database. Before estimating disk storage requirements for a database, you must know how many dimensions the database includes, the sparsity and density of the dimensions, the number of members in each dimension, and how many of the members are stored members.

➤ To calculate the disk space required for a database, perform these tasks:

1. Calculate the factors identified in "Calculating the Factors To Be Used in Sizing Disk Requirements" on page 1358.
2. Use the process described in "Estimating Disk Space Requirements for a Single Database" on page 1364 to calculate the space required for each component of a single database. If a server contains more than one database, you must perform calculations for each database.
3. Use the procedure outlined in "Estimating the Total Analytic Server Disk Space Requirement" on page 1374 to calculate the final estimate for the server.

Note: The database sizing calculations in this chapter assume an ideal scenario with an optimum database design and unlimited disk space. The amount of space required is difficult to determine precisely because most multidimensional applications are sparse.

Calculating the Factors To Be Used in Sizing Disk Requirements

Before estimating disk space requirements for a database, you must calculate the factors to be used in calculating the estimate. Later in the chapter you use these values to calculate the components of a database. For each database, you add together the sizes of its components.

Table 98 lists the sections that provide instructions to calculate these factors. Go to the section indicated, perform the calculation, then write the calculated value in the Value column.

Table 98: Factors Affecting Disk Space Requirements of a Database

Database Sizing Factor	Label	Value
"Potential Number of Data Blocks" on page 1358	DA	
"Number of Existing Data Blocks" on page 1360	DB	
"Size of Expanded Data Block" on page 1361	DC	
"Size of Compressed Data Block" on page 1363	DD	

Potential Number of Data Blocks

The potential number of data blocks is the maximum number of data blocks possible in the database.

If the database is already loaded, you can see the potential number of blocks on the Statistics tab of the Database Properties dialog box of Administration Services.

If the database is not already loaded, you must calculate the value.

➤ To determine the potential number of data blocks, assume that data values exist for all combinations of stored members.

1. Using Table 99 on page 1359 as a worksheet, list each sparse dimension and its number of stored members. If there are more than seven sparse dimensions, list the dimensions elsewhere and include all sparse dimensions in the calculation.

   The following types of members are not stored members:

   - Members from attribute dimensions
   - Shared members
   - Label Only members
   - Dynamic Calc members (Dynamic Calc And Store members are stored members)

2. Multiply the number of stored members of the first sparse dimension (line a.) by the number of stored members of the second sparse dimension (line b.) by the number of stored members of the third sparse dimension (line c.), and so on. Write the resulting value to the cell labeled DA in Table 98 on page 1358.

   ```
   a * b * c * d * e * f * g (and so on)
   = potential number of blocks
   ```

Table 99: List of Sparse Dimensions with Numbers of Stored Members

Enter Sparse Dimension Name	Enter Number of Stored Members
	a.
	b.
	c.
	d.
	e.
	f.
	g.

Example

The Sample Basic database contains the following sparse dimensions:

- Product (19 stored members)
- Market (25 stored members)

Therefore, there are 19 * 25 = 475 potential data blocks.

Number of Existing Data Blocks

As compared with the potential number of blocks, the term existing blocks refers to those data blocks that Analytic Services actually creates. For Essbase to create a block, at least one value must exist for a combination of stored members from sparse dimensions. Because many combinations can be missing, the number of existing data blocks is usually much less than the potential number of data blocks.

➤ To see the number of existing blocks for a database that is already loaded, look for the number of existing blocks on the Statistics tab of the Database Properties dialog box of Administration Services. Write the value in the cell labeled DB in Table 98 on page 1358.

If the database is not already loaded, you must estimate a value.

➤ To estimate the number of existing data blocks, perform these tasks:

1. Estimate a database density factor that represents the percentage of sparse dimension stored-member combinations that have values.
2. Multiply this percentage against the potential number of data blocks and write the number of actual blocks to the cell labeled DB in Table 98 on page 1358.

```
number of existing blocks
= estimated density
* potential number of blocks
```

Example

The following three examples show different levels of sparsity and assume 100,000,000 potential data blocks:

- Extremely sparse: Only 5 percent of potential data cells exist.

```
.05 (estimated density)
* 100,000,000 (potential blocks)
= 5,000,000 existing blocks
```

- Sparse: 15 percent of potential data cells exist.

```
.15 (estimated density)
* 100,000,000 (potential blocks)
= 15,000,000 existing blocks
```

- Dense: 50 percent of potential data cells exist.

```
.50 (estimated density)
* 100,000,000 (potential blocks)
= 50,000,000 existing blocks
```

Size of Expanded Data Block

The potential, expanded (uncompressed) size of each data block is based on the number of cells in a block and the number of bytes used for each cell. The number of cells in a block is based on the number of stored members in the dense dimensions. Analytic Services uses eight bytes to store each intersecting value in a block.

➤ To see the number of existing blocks for a database that is already loaded, look for the size of an expanded data block on the Statistics tab of the Database Properties dialog box of Administration Services.

If the database is not already loaded, you must estimate the value.

➤ To determine the size of an expanded data block, perform these tasks:

1. Using Table 100 on page 1362 as a worksheet, enter each dense dimension and its number of stored members. If there are more than seven dense dimensions, list the dimensions elsewhere and include all dense dimensions in the calculation.

   The following types of members are not stored members:

   - Members from attribute dimensions
   - Shared members
   - Label Only members
   - Dynamic Calc members (Dynamic Calc and Store members are stored members.)

2. Multiply the number of stored members of the first dense dimension (line a) by the number of stored members of the second dense dimension (line b) by the number of stored members of the third dense dimension (line c), and so on, to determine the total number of cells in a block.

   ```
   a * b * c * d * e * f * g (and so on)
   = the total number of cells
   ```

3. Multiply the resulting number of cells by 8 bytes to determine the expanded block size. Write the resulting value to the cell labeled DC in Table 98 on page 1358.

   ```
   (Total number of cells) * 8 bytes per cell
   = expanded block size
   ```

Table 100: Determining the Size of a Data Block

Enter Dense Dimension Name	Number of Stored Members
	a.
	b.
	c.
	d.
	e.

Table 100: Determining the Size of a Data Block

Enter Dense Dimension Name	Number of Stored Members
	f.
	g.

Example

The Sample Basic database contains the following dense dimensions:

- Year (12 stored members)
- Measures (8 stored members)
- Scenario (2 stored members)

Perform the following calculations to determine the potential size of a data block in Sample Basic:

```
12 * 8 * 2 = 192 data cells

192 data cells
* 8 bytes
= 1,536 bytes (potential data block size)
```

Size of Compressed Data Block

Compression affects the actual disk space used by a data file. The four types of compression, bitmap, run-length encoding (RLE), zlib, and index-value affect disk space differently. For a comprehensive discussion of data compression unrelated to estimating size requirements, see "Data Compression" on page 1044.

If you are not using compression or if you have enabled RLE compression, skip this calculation and proceed to "Stored Data Files" on page 1366.

Note: Due to sparsity also existing in the block, actual (compressed) block density varies widely from block to block. The calculations in this discussion are only for estimation purposes.

➤ To calculate an average compressed block size when bitmap compression is enabled, perform the following tasks:

1. Determine an average block density value.
   - If the database is already loaded, you can see the size of an expanded data block on the Statistics tab of the Database Properties dialog box of Administration Services. Use the value that is displayed for Block Density.
   - If you want to estimate block density prior to loading data, estimate the ratio of existing data values to potential data values.
2. To determine the compressed block size, perform the following calculation and write the resulting block size to the cell labeled DD in Table 98 on page 1358.

   ```
   expanded block size * block density
   = Compressed block size
   ```

Example

Assume an expanded block size of 1,536 bytes and a block density of 25% (.25):

```
1,536 bytes
* .25
= 384 bytes (compressed block size)
```

Estimating Disk Space Requirements for a Single Database

To estimate the disk-space requirement for a database, make a copy of Table 101 or use a separate sheet of paper as a worksheet for a single database. If multiple databases are on a server, repeat this process for each database. Write the name of the database on the worksheet.

Each row of this worksheet refers to a section that describes how to size that component. Perform each calculation and write the results in the appropriate cell in the Size column. The calculations use the factors that you wrote in Table 98 on page 1358.

Table 101: Worksheet for Estimating Disk Requirements for a Database

Database Name:	
Database Component	**Size**
"Stored Data Files" on page 1366	DE
"Index Files" on page 1368	DF
"Fragmentation Allowance" on page 1369	DG
"Outline" on page 1369	DH
"Work Areas" on page 1372 (sum of DE through DH)	DI
"Linked Reporting Objects Considerations" on page 1373, if needed	DJ
Total disk space required for the database. Total the size values from DE through DJ and write the result to Table 102 on page 1375.	

After writing all the sizes in the Size column, add them together to determine the disk space requirement for the database. Add the database name and size to the list in Table 102 on page 1375. Table 102 is a worksheet for determining the disk space requirement for all databases on the server.

Repeat this exercise for each database on the server. After estimating disk space for all databases on the server, proceed to "Estimating the Total Analytic Server Disk Space Requirement" on page 1374.

The following sections describe the calculations to use to estimate components that affect the disk-space requirements of a database.

Stored Data Files

The size of the stored database depends on whether or not the database is compressed and the compression method chosen for the database. Analytic Services provides five compression-method options: bitmap, run-length encoding (RLE), zlib, index-value, and none.

Calculating the size of a compressed database is complex for a number of reasons including the following:

- The compression method used can vary by block.
- In some situations Analytic Services chooses what it considers the best method at the block level.

For a comprehensive discussion of data compression unrelated to estimating size requirements, see "Data Compression" on page 1044. The calculations in this discussion are for estimation purposes only.

The calculation for the space required to store the compressed data files (`ess`*`xxxxx`*`.pag`) uses the following factors:

- Number of existing blocks (value DB from Table 98 on page 1358)
- Fixed-size overhead (72 bytes per block)
- Size of expanded data block (value DC from Table 98 on page 1358)
- Database density: the percentage of sparse dimension stored-member combinations that have values (To calculate, see "Number of Existing Data Blocks" on page 1360.)

Calculations for No Compression

➤ To calculate database size when the compression option is none, use the following formula:

```
Number of blocks * (72 bytes + size of expanded data block)
```

Write the result in cell labeled DE in Table 101 on page 1365. Proceed to "Index Files" on page 1368.

Calculations for Compressed Databases

Because the compression method used can vary per block, the following calculation formulas are, at best, general estimates of the database size.

Bitmap Compression

➤ To estimate database size when the compression option is bitmap, use the following formula:

```
Number of blocks
* (72 bytes
+ size of expanded data block/64)
```

Write the result in cell labeled DE in Table 101 on page 1365. Proceed to "Index Files" on page 1368.

Index-Value Compression

➤ To estimate database size when the compression option is Index-value, use the following formula:

```
Number of blocks * (72 bytes
+ (1.5 * database density * expanded data block size)
```

Write the result in cell labeled DE in Table 101 on page 1365. Proceed to "Index Files" on page 1368.

RLE Compression

➤ To estimate database size when the compression option is RLE, use the formula for calculating "Bitmap Compression" on page 1367.

When the compression method is RLE, Analytic Services automatically uses the bitmap or Index-value method for a block if it determines better compression can be gained. Estimating using the bitmap calculation estimates the maximum size.

Write the result in cell labeled DE in Table 101 on page 1365. Proceed to "Index Files" on page 1368.

zlib Compression

➤ To estimate database size when the compression option is zlib, use the formula for calculating "Bitmap Compression" on page 1367:

It is very difficult to determine the size of a data block when zlib compression is used. Individual blocks could be larger or smaller than if compressed using other compression types. Calculating using the bitmap compression formula at least provides an approximation to use for this exercise.

Write the result in cell labeled DE in Table 101 on page 1365. Proceed to "Index Files" on page 1368.

Index Files

The calculation for the space required to store the index files (`ess`*`xxxxx`*`.ind`) uses the following factors:

- Number of existing blocks (value DB from Table 98 on page 1358)
- 112 bytes—the size of an index entry

To calculate the total size of a database index, including all index files, perform the following calculation. Write the size of the compressed data files to the cell labeled DF in Table 101 on page 1365.

```
number of existing blocks * 112 bytes = the size of database index
```

Example

Assume a database with 15,000,000 blocks.

```
15,000,000 blocks
* 112
= 1,680,000,000 bytes
```

Note: If the database is already loaded, select the Storage tab on the Database Properties window.

Fragmentation Allowance

If you are using bitmap or RLE compression, a certain amount of fragmentation occurs. The amount of fragmentation is based on individual database and operating system configurations and cannot be precisely predicted.

As a rough estimate, calculate 20% of the compressed database size (value DE from Table 101 on page 1365) and write the result to the cell labeled DG in the same table.

Example

Calculating fragmentation allowance assuming a compressed database size of 5,769,000,000 bytes:

```
5,769,000,000 bytes
* .2
= 1,153,800,000 bytes
```

Outline

The space required by an outline can have two components.

- The main area of the outline is a component of both disk and memory space requirements and is calculated using the following factors:
  - The number of members in the outline
  - The length, in characters, of member names and aliases
- The attribute association area of an outline is a component only of disk space and is calculated using the following factors:
  - The number of members in each base dimension
  - The number of members in each attribute dimension

➤ To estimate the size of the outline file, perform these tasks:

1. Estimate the main area of the outline by multiplying the number of members by a name-length factor between 350 and 450 bytes.

   If the database includes few aliases or very short aliases and short member names, use a smaller number within this range. If you know that the member names or aliases are very long, use a larger number within this range.

Because the name-length factor is an estimated average, the following formula provides only a rough estimate of the main area of the outline.

```
number of members
* name-length factor
= size of main area of outline
```

Note: See Appendix A, "Limits," for the maximum sizes for member names and aliases.

For memory space requirements calculated later in this chapter, use the size of the main area of the outline.

2. For disk space requirements, if the outline includes attribute dimensions, calculate the size of the attribute association area for the database. Calculate the size of this area for each base dimension. Multiply the number of members of the base dimension by the sum of the count of members of all attribute dimensions associated with the base dimension, and then divide by 8.

   Note: Within the count of members, do not include Label Only members and shared members.

   ```
   (number of base-dimension members
   * sum of count of attribute-dimension members)/8
   = size of attribute association area for a base dimension
   ```

3. Sum the attribute association areas of each dimension to determine the total attribute association area for the outline.

4. For the total disk space required for the outline, add together the main outline area and the attribute association area, and write the result of this calculation to the cell labeled DH in Table 101 on page 1365.

   ```
   main area of outline + total attribute association area
   ```

Example

Assume the outline has the following characteristics:

- 26,000 members
- A name-length factor of 400 bytes
- A base dimension Product (23,000 members—excluding Label Only members and shared members) with two attribute dimensions associated with it—Ounces (20 members) and Pkg Type (50 members)

- A base dimension Market (2,500 members—excluding Label Only members and shared members) with one associated attribute dimension, Population (12 members)

Perform the following calculations:

1. Calculate the main area of the outline:

```
name-length factor of 400 bytes
* 26,000 members
= 10,400,000 bytes
```

2. Calculate the attribute association areas:

   - For the base dimension Product:

```
(23,000 * (20 + 50)) bits
/ 8 bits per byte
= 201,250 bytes
```

   - For the base dimension Market:

```
(2,500 * 12) bits
/ 8 bits per byte
= 3,750 bytes
```

3. Sum these areas for the total attribute association area for the database:

```
201,250 bytes + 3,750 bytes = 205,000 bytes
```

4. For a total estimate of outline disk space, add the main area of the outline and the total attribute association area:

```
10,400,000 bytes
+ 205,000 bytes
= 10,605,000 bytes (outline disk space requirement)
```

Note: Do not use this procedure to calculate outline memory space requirements. Use the process described in "Outline Size Used in Memory" on page 1381.

Work Areas

Three different processes create temporary work areas on the disk:

- For recovery purposes, Analytic Services maintains a data recovery area on the disk. The size of this area increases until the database is restructured.
- During restructuring, Analytic Services uses a restructuring work area on the disk.
- During migration from prior releases of Analytic Services, for recovery purposes, Analytic Services creates a copy of the database in a migration work area.

To create these temporary work areas, Analytic Services may require disk space equal to the size of the entire database. Restructuring and migration need additional work space the size of the outline. Because none of these activities occur at the same time, a single allocation can represent all three requirements.

To calculate the size of a work area used for restructuring, migration, and recovery, calculate the sum of the sizes of the following database components from Table 101 on page 1365:

- Compressed data files (value DE)
- Index files (value DF)
- Fragmentation allowance (value DG)
- Outline (value DH)

Use the following formula to calculate the size of the work area:

```
work area = size of compressed data files
+ size of index files
+ fragmentation allowance
+ outline size
```

Write the result of this calculation to the cell labeled DI in Table 101 on page 1365.

Linked Reporting Objects Considerations

You can use the Linked Reporting Objects (LROs) feature to associate objects with data cells. The objects can be flat files, HTML files, graphics files, and cell notes. For a comprehensive discussion of linked reporting objects, see Chapter 11, "Linking Objects to Analytic Services Data."

Two aspects of LROs affect disk space:

- The size of the object. Because Analytic Services copies files used as LROs to the server, you must know the combined size of all files you are using as LROs.

  Note: You can set a limit on the size of a linked object, if the linked object is a file (as opposed to a cell note). For a discussion of why and how to limit LRO file sizes, see "Limiting LRO File Sizes" on page 1249.

- The size of the LRO catalog, where the Analytic Services Kernel stores information about LROs. Cell notes and URLs are also stored in the catalog. A catalog entry is stored as an index page. For every catalog entry, Analytic Services uses 8 KB.

➤ To estimate the disk space requirements for linked reporting objects, perform the following tasks:

1. Estimate the size of the objects. If a limit is set, multiply the number of LROs by that limit. Otherwise, sum the size of all anticipated LROs.
2. Size the LRO catalog. Multiply the total number of LROs by 8192 bytes.
3. Add together the two areas and write the result of this calculation to the cell labeled DJ in Table 101 on page 1365.

Example

Assume the database uses 1500 LROs which are composed of the following:

- 1000 URLs, at a maximum of 512 bytes each
- 500 cell notes

Perform the following calculations:

1. Multiply 1000 * 512 bytes for 512,000 bytes maximum required for the stored URLs.
2. Calculate the size of the LRO catalog. Multiply 1500 total LROs * 8192 bytes = 12,288,000 bytes.
3. Add together the two areas; for example:

```
512,000 bytes
+ 12,288,000 bytes
= 12,800,000 bytes total LRO disk space requirement
```

Estimating the Total Analytic Server Disk Space Requirement

The earlier calculations in this chapter estimate the data storage requirement for a single database. Often, more than one database resides on the server.

In addition to the data storage required for each database, the total Analytic Services data storage requirement on a server includes Analytic Services software. Allow approximately 200 MB (209,715,200 bytes) for the base installation of Analytic Services software and sample applications. The allowance varies by platform and file management system. For details, see the *Essbase Analytic Services Installation Guide*.

➤ To estimate the total server disk space requirement, perform the following tasks:

1. In the worksheet in Table 102 on page 1375, list the names and disk space requirements that you calculated for each database.
2. Sum the database requirements and write the total in bytes in the cell labeled DL.
3. In the cell labeled DM, write the appropriate disk space requirement for the software installed on the server; for example, 209,715,200 bytes.
4. For the total server disk space requirement in bytes, sum the values in cells DL and DM. Write this value in the cell labeled DN.
5. Convert the total in cell DN to megabytes (MB) by dividing the value by 1,048,576 bytes. Write this value in the cell labeled DO.

Table 102: Worksheet for Total Server Disk Space Requirement

List of Databases (From Table 101 on page 1365)	**Size**
a.	
b.	
c.	
d.	
e.	
f.	
g.	
Sum of database disk sizes a + b + c + d + e + f + g	DL:
Size in bytes for Analytic Services server software	DM:
Total Analytic Server disk requirement in bytes: DL + DM	DN:
Total Analytic Server disk requirement in megabytes (MB): DN divided by 1,048,576 bytes	DO:

Estimating Memory Requirements

The minimum memory requirement for running Analytic Services is 64 MB. On UNIX systems, the minimum requirement is 128 MB. Based on the number of applications and databases and the database operations on the server, the amount of memory you require may be more.

Analytic Services provides a memory management feature that enables you to specify the maximum memory to be used for all server activities, or a maximum memory that can be used to manage specific applications. For additional information about this feature, see the Memory Manager Configuration section of the *Technical Reference*.

If you use the memory management feature to limit the amount of memory available to the server, you do not need to calculate a memory requirement. The total memory required on the computer is equal to the sum of the operating system

memory requirement plus the Analytic Server limit you specify in the MEMORYLIMIT configuration setting in the `config.mem` memory configuration file.

To estimate the memory required on Analytic Server, use the Worksheet for Total Memory Requirements, Table 107 on page 1392, to collect and total server memory requirements. To calculate the requirements for this worksheet, review the following topics:

- "Estimating Memory Requirements for Applications" on page 1376
- "Estimating Startup Memory Requirements for Databases" on page 1378
- "Estimating Additional Memory Requirements for Database Operations" on page 1385
- "Estimating Total Essbase Memory Requirements" on page 1392

Estimating Memory Requirements for Applications

If you use the memory management feature to control the amount of memory used by Analytic Server for all applications, do not calculate application and database memory requirements. See "Estimating Memory Requirements" on page 1375.

The approach to determining the amount of memory required for an application varies, depending on whether or not you set memory limits on individual applications. As appropriate to your individual applications, follow the instructions in the following topics:

- "Application Memory Limited by Memory Manager" on page 1377
- "Startup Memory Requirement for Applications" on page 1378

Application Memory Limited by Memory Manager

If you use the memory management feature to limit the amount of memory available for individual applications, you do not need to calculate the memory requirements for those applications. For information about setting memory maximums for individual applications, see the Memory Manager Configuration section of the *Technical Reference*.

To determine the maximum amount of memory that can be used by applications for which memory limits are established in application memory configuration files, list the applications in Table 103 and write the associated memory limit in the Maximum Size column.

Table 103: Applications With Specified Maximum Sizes

Application Name:	**Maximum Size, in Megabytes (MB)**
a.	
b.	
c.	
d.	
e.	
f.	
g.	

Total the memory values and write the result to the cell labeled ML in "Worksheet for Total Server Memory Requirement" on page 1392.

Startup Memory Requirement for Applications

For application memory use that is not controlled by Memory Manager, you need to calculate overall memory used at application startup plus the memory requirements for each database.

Each open application has the following memory requirements at startup:

- Analytic Services code and static data (10 MB). This number may be more or less, depending on the operating system used.
- **Optional:** Java Virtual Machine (2 to 4 MB), which is used for custom-defined functions. This value depends on the operating system used.

Multiply the number of applications that will be running simultaneously on the server by the appropriate startup requirement and write the resulting value to the cell labeled MM in Table 107 on page 1392. Do not include in this calculation applications for which the amount of memory used is controlled by Memory Manager.

Estimating Startup Memory Requirements for Databases

Calculate memory requirements for each database on Analytic Server.

Note: Do not include in this calculation databases within applications for which you use the memory management feature to limit the amount of memory available to them.

For each database, make a copy of Table 107 or use a separate sheet of paper as a worksheet for a single database. If multiple databases are on Analytic Server, repeat this process for each database. Write the name of the database on the worksheet.

Each row links to information that describes how to size that component. Perform each calculation and note the results in the appropriate cell in the Size column. Some calculations use the factors that you wrote in Table 105 on page 1380. After filling in all the sizes in the Size column, add them together to determine the memory requirement for that database.

After estimating disk space for all databases on the server, proceed to "Estimating the Total Analytic Server Disk Space Requirement" on page 1374.

Table 104: Worksheet for Estimating Memory Requirements for a Database

Database Name:	**Size in Megabytes (MB)**
Memory Requirement:	
Startup requirements per database:	
• Database outline. (See "Outline Size Used in Memory" on page 1381.)	MA:
• Index cache. (See "Sizing Caches" on page 1128.)	MB:
• Cache-related overhead. (See "Cache-Related Overhead" on page 1382.)	MC:
• Area for data structures. (See "Memory Area for Data Structures" on page 1384.)	MD:
Operational Requirements:	
• Memory used for data retrievals. (See "Estimating Additional Memory Requirements for Data Retrievals" on page 1385.)	ME:
• Memory used for calculations. (See "Estimating Additional Memory Requirements for Calculations" on page 1391.)	MF:
Summarize the size values from MA through MF for an estimate of the total memory required for a database.	MG:
Divide the value from MG by 1,048,576 bytes for the total database memory requirement in megabytes (MB).	MH:

In Table 107 on page 1392, enter the name of the database and the total memory requirement in megabytes, MH.

Factors to Be Used in Sizing Memory Requirements

Before you start the estimate, calculate factors to be used in calculating the estimate.

Table 105 on page 1380 lists sizing factors with references to sections in this chapter and other chapters that provide information to determine these sizes. Go to the section indicated, perform the calculation, then return to Table 105 and write the size, in bytes, in the Value column of this table.

Later in this chapter, you can refer to Table 105 for values to use in various calculations.

Table 105: Factors Used to Calculate Database Memory Requirements

Database Sizing Factor	Value
The number of cells in a logical block. (See "The Number of Cells in a Logical Block" on page 1383.)	MI:
The number of threads allocated through the SERVERTHREADS ESSCMD. (See the *Technical Reference*.)	MJ:
Potential stored-block size. (See "Size of Expanded Data Block" on page 1361.)	MK:

The calculations in this chapter do not account for other factors that affect how much memory is used. The following factors have complex implications and are not included within the discussion:

- Cache memory locking. Whether or not cache memory locking is enabled affects how the operating system and Essbase manage memory. For an explanation of why and how to enable cache memory locking, see "Deciding Whether to Use Cache Memory Locking" on page 1127.
- Different operation types and their associated cache allocations. Data load, data retrieval, and calculation operations set aside memory for the data file, data, and calculation caches, plus some overhead associated with the caches.
- The sizes of the retrieval buffer and the retrieval sort buffer. See "Changing Buffer Size" on page 1244 for a discussion of the significance of the size of the retrieval buffer and the retrieval sort buffer.

- Database workload; for example, the complexity of a calculation script or the number and complexity of data queries.
- The number of data blocks defined using the CALCLOCKBLOCK setting in the `essbase.cfg` file in combination with the SET LOCKBLOCK setting, which specifies which CALCLOCKBLOCK setting to use.
- The number of Dynamic Calc members in the outline, including members of attribute dimensions.
- The isolation level settings.
- Synchronization points.
- Use of triggers.
- MDX implications.

Outline Size Used in Memory

The attribute association area included in disk space calculations is not a sizing factor for memory. Calculate only the main area of the outline.

For memory size requirements, outline size is calculated using the following factors:

- The number of members in the outline
- The length, in characters, of member names and aliases

➤ To calculate the outline memory requirement, multiply the number of members by a name-length factor between 300 and 400 bytes and write the result to the cell labeled MA in Table 104 on page 1379.

If the database includes few aliases or very short aliases and short member names, use a smaller number within the 300–400 byte range. If you know that the names or aliases are very long, use a larger number within this range.

Because the name-length factor is an estimated average, the following formula provides only a rough estimate of the main area of the outline:

```
memory size of outline
= number of members
* name-length factor
```

Note: See Appendix A, "Limits," for the maximum sizes for member names and aliases.

Example

Assuming the outline has 26,000 members and a median name-length, use the following calculation to estimate the outline size used in memory:

```
26,000 members
* 350 bytes per member
= 9,100,000 bytes
```

Index Cache

At startup, Essbase sets aside memory for the index cache, the size of which can be user-specified. To determine the size of the index cache, see "Sizing Caches" on page 1128 and write the size in the cell labeled MB in Table 104 on page 1379.

Cache-Related Overhead

Analytic Services uses additional memory while it works with the caches.

The calculation for this cache-related overhead uses the following factors:

- Index cache (value MB from Table 104 on page 1379)
- The number of server threads (value MJ from Table 105 on page 1380)

➤ To calculate the cache-related overhead at startup, perform the following tasks:

1. Calculate half the index cache size, in bytes.

   ```
   index cache size
   * .5
   = index cache-related overhead
   ```

2. Calculate additional cache overhead in bytes using the following formula:

   ```
   ((# of server threads allocated to the Analytic Server process
   * 3)
   * 256)
   + 5242880 bytes
   = additional cache overhead
   ```

3. Sum the index cache overhead plus the additional cache overhead. Write the result to the cell labeled MC in Table 104 on page 1379.

```
cache-related overhead
= index cache-related overhead
+ additional cache overhead
```

The Number of Cells in a Logical Block

The term logical block refers to an expanded block in memory.

➤ To determine the cell count of a logical block, multiply together all members of each dense dimension (including Dynamic Calc and Dynamic Calc and Store members but excluding Label Only and shared members).

1. Using Table 106 on page 1383 as a worksheet, enter each dense dimension and its number of members excluding Label Only and shared members. If there are more than seven dense dimensions, list the dimensions elsewhere and include all dense dimensions in the calculation.
2. Multiply the number of members of the first dense dimension (line a.) by the number of members of the second dense dimension (line b.) by the number of members of the third dense dimension (line c.), and so on, to determine the total number of cells in a logical block. Write the result to the cell labeled MI in Table 105 on page 1380.

```
a * b * c * d * e * f * g = the total number of cells
```

Table 106: Determining the Number of Cells in a Logical Block

Enter Dense Dimension Name	Number of Members
	a.
	b.
	c.
	d.
	e.
	f.
	g.

Example

Excluding Label Only and shared members, the dense dimensions in Sample Basic contain 17 (Year), 14 (Measures), and 4 (Scenario) members. The calculation for the cell count of a logical block in Sample Basic is as follows:

```
17 * 14 * 4 = 952 cells
```

Memory Area for Data Structures

At application startup time, Analytic Services sets aside an area of memory based on the following factors:

- The number of members in the outline
- The number of cells in a logical block (value MI in Table 105 on page 1380)
- The number of threads on the server (value MJ in Table 105)

➤ To calculate the data structure area in memory, perform the following tasks:

1. Use the following formula to calculate the size in bytes:

   ```
   Number of threads
   * ((Number of members in the outline * 26 bytes)
   + (Logical block cell count * 36 bytes))
   ```

2. Write the result to the cell labeled MD in Table 104 on page 1379.

Example

Assuming 20 threads for the Sample Basic database, the startup area in memory required for data structures is calculated as follows:

```
20 threads
* ((79 members * 26 bytes) + (952 cells * 36 bytes))
= 726,520 bytes 726,520 bytes
/ 1,048,576 bytes = .7 MB
```

Estimating Additional Memory Requirements for Database Operations

In addition to startup memory requirements, operations such as queries and calculations require additional memory. Because of many variables, the only way to estimate memory requirements of operations is to run sample operations and monitor the amount of memory used during these operations. This topic provides guidelines for the following estimation tasks:

- "Estimating Additional Memory Requirements for Data Retrievals" on page 1385
- "Estimating Additional Memory Requirements Without Monitoring Actual Queries" on page 1389
- "Estimating Additional Memory Requirements for Calculations" on page 1391

Estimating Additional Memory Requirements for Data Retrievals

Analytic Services processes requests for database information (queries) from a variety of sources. For example, Analytic Services processes queries from the Spreadsheet Add-in and from Report Writer. Analytic Services uses additional memory when it retrieves the data for these queries, especially when Analytic Services must perform dynamic calculations to retrieve the data. This section describes Analytic Services memory requirements for query processing.

Analytic Services is a multithreaded application in which queries get assigned to threads. Threads are automatically created when Analytic Services is started. In general, a thread exists until you shut down Analytic Server. For an explanation of how Analytic Services uses threads, see "Multithreading" on page 913.

As Analytic Services processes queries, it cycles through the available threads. For example, assume 20 threads are available at startup. As each query is processed, Analytic Services assigns each succeeding query to the next sequential thread. After it has assigned the 20th thread, Analytic Services cycles back to the beginning, assigning the 21st query to the first thread.

While processing a query, a thread allocates some memory, and then releases most of it when the query is completed. Some of the memory is released to the operating system and some of it is released to the dynamic calculator cache for the database being used. However, the thread holds on to a portion of the memory for possible

use in processing subsequent queries. As a result, after a thread has processed its first query, the memory held by the thread is greater than it was when Analytic Services first started.

Analytic Services uses the maximum amount of memory for query processing when both of these conditions are true:

- The maximum number of simultaneous queries that will occur are being processed.
- All threads have been assigned to at least one query by Analytic Services.

In the example where 20 threads are available at startup, the maximum amount of memory is used for queries when at least 20 queries have been processed and the maximum number of simultaneous queries are in process.

Calculating the Maximum Amount of Additional Memory Required

➤ To estimate query memory requirements by observing actual queries, perform the following tasks:

1. Observe the memory used during queries.
2. Calculate the maximum possible use of memory for query processing by adding together the memory used by queries that will be run simultaneously, then add the extra memory that had been acquired by threads that are now waiting for queries.

Use the following variables when you calculate the formula in "Estimating the Maximum Memory Usage for A Query Before and After Processing" on page 1387:

- Total number of threads (*Total#Threads*)
- Maximum memory usage for any query *during* processing (*MAXAdditionalMemDuringP*)
- Maximum number of possible concurrent queries (*Max#ConcQueries*)
- Maximum memory usage for any query *after* processing (*MAXAdditionalMemAfterP*)

Determining the Total Number of Threads

The potential number of threads available is based on the number of licensed ports that are purchased. The actual number of threads available depends on settings you define for the Agent or the server. Use the number of threads on the system as the value for *Total#Threads* in later calculations.

Estimating the Maximum Number of Concurrent Queries

Determine the maximum number of concurrent queries and use this value for *Max#ConcQueries* in later calculations. This value cannot exceed the value for *Total#Threads.*

Estimating the Maximum Memory Usage for A Query Before and After Processing

The memory usage of individual queries depends on the size of each query and the number of data blocks that Analytic Services needs to access to process each query. To estimate the memory usage, calculate the additional memory Analytic Services uses during processing and after processing each query.

Decide on several queries that you expect to use the most memory. Consider queries that must process large numbers of members; for example, queries that perform range or rank processing.

➤ To estimate the memory usage of a query, perform the following tasks:

1. Turn the dynamic calculator cache off by setting the `essbase.cfg` setting DYNCALCACHEMAXSIZE to 0 (zero). Turning off the dynamic calculator cache enables measurement of memory still held by a thread by ensuring that, after the query is complete, the memory used for blocks during dynamic calculations is released by the ESSSVR process to the operating system.
2. Start the Analytic Services application.
3. Using memory monitoring tools for the operating system, note the memory used by Analytic Server *before* processing the query. Use the value associated with the ESSSVR process.

   Use this value for *MemBeforeP.*
4. Run the query.

5. Using memory monitoring tools for the operating system, note the peak memory usage of Analytic Server *while* the query is processed. This value is associated with the ESSSVR process.

   Use this value for *MemDuringP*.

6. Using memory monitoring tools for the operating system, *after* the query is completed, note the memory usage of Analytic Services. This value is associated with the ESSSVR process.

   Use this value for *MemAfterP*.

7. Calculate the following two values:

   - Additional memory used while Analytic Services processes the query (*AdditionalMemDuringP*):

     ```
     AdditionalMemDuringP = MemDuringP - MemBeforeP
     ```

   - Additional memory used after Analytic Services has finished processing the query (*AdditionalMemAfterP*):

     ```
     AdditionalMemAfterP = MemAfterP - MemBeforeP
     ```

8. When you have completed the above calculations for all the relevant queries, compare all results to determine the following two values:

   - The maximum *AdditionalMemDuringP* used by a query: (*MAXAdditionalMemDuringP*)
   - The maximum *AdditionalMemAfterP* used by a query: (*MAXAdditionalMemAfterP*)

9. Insert the two values from step 7 into the formula in the following statement.

   The amount of additional memory required for data retrievals will not exceed the following:

   ```
   Max#ConcQueries
   * MAXAdditionalMemDuringP
   + (Total#Threads - Max#ConcQueries)
   * MAXAdditionalMemAfterP
   ```

   Write the result of this calculation, in bytes, to the cell labeled ME in Table 104 on page 1379.

Because this calculation method assumes that all of the concurrent queries are maximum-sized queries, the result may exceed your actual requirement. It is difficult to estimate the actual types of queries that will be run concurrently.

To adjust the memory used during queries, you can set values for the retrieval buffer and the retrieval sort buffer. For a review of methods, see "Setting the Retrieval Buffer Size" on page 1244 and "Setting the Retrieval Sort Buffer Size" on page 1245.

Estimating Additional Memory Requirements Without Monitoring Actual Queries

If you cannot perform this test with actual queries, you can calculate a very rough estimate for operational query requirements. Requirements for each retrieval vary considerably. As a generalization, this estimate uses the following fixed factors:

- The size of the retrieval buffer (10,240 bytes)
- If sorting is involved in the query, the size of the retrieval sort buffer (20.480 bytes)
- The size of the buffer which holds formatting information (140 KB, which rounds up to 144,000 bytes)
- Report Writer factors:
  - 40 bytes per selected member in the retrieval
  - 8 bytes per member of the largest dimension

This estimate also uses the following variables:

- The amount of memory used for dynamically calculated values, which is based on the following numbers:
  - The number of blocks specified by the SET LOCKBLOCK command
  - The number of cells in a logical block, which includes Dynamic Calc members. See value MH. in Table 105 on page 1380.
- The size of the data cache. See "Sizing the Data Cache" on page 1132.
- If direct I/O is used, the size of the data file cache. See "Sizing the Data File Cache" on page 1130.

- Report Writer factors:
  - The number of selected members in an average retrieval
  - The number of members in the largest dimension to calculate

You can then use the following two calculations for the memory needed for retrievals:

- Buffer and work area used in each retrieval:

```
retrieval buffer (10,240 bytes)
+ retrieval sort buffer (20,480 bytes)
+ formatting buffer (144,000 bytes)
+ dynamic calc area
+ data cache size
+ data file cache size
```

- Memory needed for estimated number of concurrent retrievals:

```
Member storage area for the largest dimension
+ (number of retrievals
* sum of buffer and work areas used in each retrieval)
```

Summarize the calculations and write the result, in bytes, to the cell labeled ME in Table 104 on page 1379.

Example

To estimate the maximum memory needed for concurrent queries, assume the following values for this example:

- The area required by the largest dimension:

```
23,000 members * 8 bytes/member = 184,000 bytes
```

- The buffer and work area values apply for each retrieval:
  - retrieval buffer: 10,240 bytes
  - retrieval sort buffer: 20,480 bytes
  - formatting buffer: 144,000 bytes
  - dynamic calc area: 761,600 bytes

    With SET LOCKBLOCK set as 100 blocks, the calculation is:

    ```
    100 blocks * 7616-byte block size = 761,600 bytes
    ```

  - data cache size: 3072 KB which equals 3,145,728 bytes
  - data file cache size: zero. Direct I/O is not used in the example.
- The largest dimension has 23,000 members.
- The maximum number of concurrent queries is 20
- The number of selected members is generalized across all queries to be 10,000 members. The approximate memory requirement equals the following:

  ```
  10000 members * 40 bytes/member = 400,000 bytes
  ```

Estimated memory for retrievals is as follows:

```
184,000 bytes + (20 concurrent inquiries
* (10,240 bytes + 20,480 bytes + 144,000 bytes
+ 761,600 bytes + 3,145,728 bytes + 400,000 bytes))
= 75,824,960 bytes
```

Estimating Additional Memory Requirements for Calculations

For existing calculation scripts, you can use the memory monitoring tools provided for the operating system on the server to observe memory usage. Run the most complex calculation and take note of the memory usage both before and while running the calculation. Calculate the difference and use that figure as the additional memory requirement for the calculation script.

For a comprehensive discussion of calculation performance, see Chapter 54, "Optimizing Calculations."

If you cannot perform a test with a calculation script, you can calculate a very rough estimate for the operational requirement of a calculation by adding the following values:

- The size of the calculator cache. See "Sizing the Calculator Cache" on page 1133.
- The size of the data cache. See "Sizing the Data Cache" on page 1132.
- The size of the outline. For calculations, Analytic Services uses approximately 30 additional bytes of memory per member of the database outline. For an example and for information about concurrent calculations, see "Managing Caches to Improve Performance" on page 1202).

- The size of the memory area used by the blocks set aside by the SET LOCKBLOCK command. To calculate the memory requirement in bytes, multiply the specified number of data blocks by the logical size of the data blocks. For the logical block size, multiply the number of cells (value MI in Table 105 on page 1380) by 8 bytes per cell.

For the total calculation requirement, summarize the amount of memory needed for all calculations that will be run simultaneously and write that total to the cell labeled MF in Table 104 on page 1379.

Note: The size and complexity of the calculation scripts affect the amount of memory required. The effects are difficult to estimate.

Estimating Total Essbase Memory Requirements

You can use Table 107 as a worksheet on which to calculate an estimate of the total memory required on the server.

Table 107: Worksheet for Total Server Memory Requirement

Component	Memory Required, in Megabytes (MB)
Sum of memory maximums established for individual applications (See "Application Memory Limited by Memory Manager" on page 1377.)	ML
Sum of application startup memory requirements (See "Startup Memory Requirement for Applications" on page 1378.)	MM:
In rows a through g below, list concurrent databases (from copies of Table 104 on page 1379) and enter their respective memory requirements (MH) in the column to the right.	
a.	MH:
b.	MH:
c.	MH:
d.	MH:
e.	MH:
f.	MH:

Table 107: Worksheet for Total Server Memory Requirement (Continued)

Component	Memory Required, in Megabytes (MB)
g.	MH:
Operating system memory requirement	MN:
Total estimated memory requirement for the server	MO:

➤ To estimate the total Analytic Services memory requirement on a server, perform the following tasks:

1. If the memory management feature is used to limit the amount of memory used by Analytic Services, the total memory required on the computer is equal to the sum of the operating system memory requirement plus the Analytic Server limit you specify in the MEMORYLIMIT configuration setting in the `config.mem` memory configuration file. No further calculation is necessary.
2. If the memory management feature is not used to limit the amount of memory used by Analytic Services for all applications, perform the following steps:
   a. Record in the cell labeled ML the sum of memory maximums defined for individual applications, as described in topic "Application Memory Limited by Memory Manager" on page 1377.
   a. Record in the cell labeled MM the total startup memory requirement for applications, as described in topic "Startup Memory Requirement for Applications" on page 1378.
   b. List the largest set of databases that will run concurrently on the server. In the Memory Required column, for the MH value for each database, note the memory requirement estimated in the database requirements worksheet, Table 104 on page 1379.
   c. Determine the operating system memory requirement and write the value in megabytes to the cell labeled MN in Table 107.
   d. Total all values and write the result in the cell labeled MO.

e. Compare the value in MN with the total available random-access memory (RAM) on the server.

Note: In addition, be sure to consider memory requirements for client software, such as Essbase Administration Services, that may be installed on the Analytic Server computer. See the appropriate documentation for details.

If cache memory locking is enabled, the total memory requirement should not exceed two-thirds of available RAM; otherwise, system performance can be severely degraded. If cache memory locking is disabled, the total memory requirement should not exceed available RAM.

If there is insufficient memory available, you can redefine the cache settings and recalculate the memory requirements. This process can be iterative. For guidelines and considerations, see "Fine Tuning Cache Settings" on page 1144. In some cases, you may need to purchase additional RAM.

Appendix D

Using ESSCMD

The Analytic Server includes MaxL and ESSCMD, both command-line interfaces that perform operations interactively or through batch or script files.

MaxL improves upon ESSCMD in that it is a multi-dimensional access language for Analytic Services. Using MaxL, you make requests to Analytic Services using English-like statements, rather than commands. Essbase® scripts can be developed flexibly to accommodate many uses, and the MaxL language is easy to learn. MaxL is installed with a Perl module that enables you to embed its statements in Perl programs.

This appendix focuses on ESSCMD. It describes how to use ESSCMD interactive and batch processing modes and provides sample script and batch files. It includes the following topics:

For MaxL and ESSCMD syntax and usage information, see the *Technical Reference*.

For information on Administration Services and MaxL Script Editor, see the *Essbase Administration Services Online Help*.

Understanding ESSCMD

With ESSCMD, you can execute server operations at the command line, in either interactive or batch mode.

Interactive mode means entering commands at the ESSCMD command line, and receiving prompts where necessary. Interactive mode is convenient for short operations that require few commands, checking for information on the fly, and error checking.

Batch processing mode is used for automating routine server maintenance and diagnostic tasks. You can write a script or batch file and run it from the command line. Batch processing mode is convenient if you frequently use a particular series of commands, or if a task requires many commands.

Understanding Syntax Guidelines

In general, use the same syntax for entering ESSCMD commands as you do for other calculation commands. However, there are differences between ESSCMD's interactive and batch processing modes in the requirements for quotation marks and the semicolon statement terminator. Use the guidelines in this section when creating script or batch files.

Quotation Marks

Quotation marks ("") enclose character parameters and responses to commands.

- In interactive ESSCMD, using quotes is optional. Be sure to use quotes when a parameter has an embedded space; for example:

  ```
  CALC "Calc All;"
  ```

- In an ESSCMD script file, always enclose all character parameters and responses to commands in quotation marks; for example:

  ```
  LOGIN "Localhost" "user1" "Password";
  ```

- Numeric parameters and responses do not require quotation marks.
- Do not enclose quotation marks within quotation marks.

Semicolon Statement Terminator

The ; (semicolon) statement terminator signals the end of a command; for example:

```
SELECT "SAMPLE" "BASIC";
```

- In interactive ESSCMD, pressing the Enter key signals ESSCMD that the command is complete. The statement terminator is optional.
- In an ESSCMD script file, you should use the terminator, even though it is optional, if a command has many parameters. This is especially important in order to signal the end of the parameter list if some of the parameters are optional.

  If you omit some optional parameters and do not use a semicolon to end the list, ESSCMD looks for the remaining values in the next command in the file, leading to unpredictable results.

The SETAPPSTATE and SETDBSTATE commands, defined in the *Technical Reference*, are examples of commands which you should terminate with a semicolon to prevent any confusion in processing.

Note: All syntax examples in this chapter use quotation marks and semicolon terminators.

Running ESSCMD on Different Operating System Platforms

ESSCMD operates independently of any Analytic Services client interface, including Administration Services, Spreadsheet Add-in, or custom-built application programs.

ESSCMD is available on the platforms listed in the *Essbase Analytic Services Installation Guide*, in the server platform system requirements section.

Canceling ESSCMD Operations

When running ESSCMD, you can cancel an asynchronous operation, such as a calculation, export, or restructure operation, by pressing and holding the Esc key until ESSCMD responds.

Referencing Files

Some commands require that you precede object or file names with a numeric parameter, from 1 to 4, that tells Analytic Services where to look for the object or file. The parameter directs ESSCMD to look for files in other applications, databases, or systems.

The following table lists each value for the numeric parameter (*Number*), the file location to which it applies, and the information that ESSCMD requests when you use each parameter setting. *appName* is the application name and *dbName* is the database name.

Number	File	Analytic Services prompts user for:
1	Local or client-based file	• Windows: Files in the `\essbase\client\`*`appname`*`\`*`dbName`* directory • UNIX: Files in the `essbase/client/`*`appName/dbName`* directory
2	Remote or server-based file	• Windows: Files in the `\essbase\app\`*`appName`*`\`*`dbName`* directory • UNIX: Files in the `essbase/app/`*`appName/dbName`* directory
3	File	Fully-qualified path to the file, unless file is in the current ESSCMD directory
4	SQL table	Full network and database information for the SQL table

For example, the LOADDATA command can load a data file that resides on the client or Analytic Server. The command requires the numeric parameter to tell Analytic Services where to look for the data file. This example causes ESSCMD to prompt for the fully-qualified path name of the file to load:

```
LOADDATA 3
```

File extensions are usually optional in both interactive and batch processing modes, except when using commands that require a numeric parameter that indicates the location of files:

- If you use file option 3 (File), you must enter the file extension in both interactive and batch processing modes.
- If the object is in the directory from which you started ESSCMD, you do not need to enter a path.

Accessing Multiple Databases

Because ESSCMD supports multiple login instances on Analytic Server, you can access more than one database in a single session. Even when you log in to multiple databases, you use only one port on your server license.

Considering Case-Sensitivity

The Analytic Server creates application and database names exactly as the user specifies. Case is not changed for any platform.

For backward compatibility, the Analytic Server searches for existing application and database names using the exact case first. However, if it cannot find the file, Analytic Server searches all possible case combinations to find the existing application and database names.

Analytic Services does not allow you to create application and database names that differ only in case. For example, Analytic Services would display an error message if you tried to create an application MyData if you already had an application mYdATA.

You can choose to make member names case sensitive.

➤ To make a member case-sensitive, see “Making Members Case-Sensitive” in the *Essbase Administration Services Online Help*.

Getting Help

To display a list of all currently available ESSCMD commands, enter HELP?. To see documented descriptions and syntax for individual ESSCMD commands, see the online *Technical Reference*, located in the `ESSBASE/DOCS` directory.

Preparing to Start ESSCMD

Before you start ESSCMD, make sure that the following items are properly installed and running:

- Analytic Services
- Communications protocol (Named Pipes or TCP/IP) on Analytic Services

For information on protocols supported by Analytic Services, see the *Essbase Analytic Services Installation Guide*.

Starting and Quitting ESSCMD

The Analytic Services installation places the `esscmd.exe` and `esscmd.hlp` files (`esscmd` and `esscmd.hlp` on UNIX platforms) in the `bin` directory of your application server.

Once you start the application, a command prompt like this one displays:

`:::[n]->`

where *n* is the value of the active login instance. Each subsequent, successful login increments this value by one. When you start ESSCMD, the instance number is zero (0).

Note: Use the SETLOGIN command to toggle between active login instances. Use the LISTLOGINS command to view active login instances.

➤ To start ESSCMD, do one of the following:

- Enter `ESSCMD` at the operating system command prompt. ESSCMD runs within the operating system command line window.
- Run `esscmd.exe` (`esscmd` on UNIX platforms), located in the `bin` directory

➤ To quit ESSCMD, Enter EXIT at the prompt and press Enter. ESSCMD disconnects from the application server, and terminates the session.

Using Interactive Mode

In interactive mode, you enter commands and respond to prompts. This is useful when you are performing simple tasks that require few commands. If you are performing more complex tasks that require many commands, consider creating a script file or batch file; see "Using Script and Batch Files for Batch Processing" on page 1403 for information and instructions.

For syntax conventions when working in interactive mode, see "Understanding Syntax Guidelines" on page 1396.

Logging on to Analytic Server

After starting ESSCMD, you must connect to Analytic Server so that you can enter commands. Follow these steps:

1. At the ESSCMD prompt, log in to the server with the LOGIN command. For more information about this command, see the online *Technical Reference*, located in the `essbase/docs` directory.
2. Enter the server name. When you connect from the server console, the server name depends on your network setup. For example, the name could be `aspen`.
3. Enter your user name.
4. Enter your password.

The ESSCMD prompt is displayed as follows:

`aspen:::`*`userName`*`[1]->`

userName is your login name.

You can enter any valid ESSCMD command. For a complete listing of commands, type HELP.

Note: To load an application into memory and select a database on the Analytic Services server, use the SELECT command to select a database from an application that resides on the server.

The ESSCMD prompt is displayed as follows:

```
aspen:appName:dbName:userName[1]->
```

appName is the name of the application. *dbName* is the name of the database to which you are connected.

Entering Commands

There are two ways to enter commands in interactive mode. Choose either of the following methods to enter commands:

- Type the command and press Enter.

  ESSCMD prompts you for each of the command parameters. For example, the SELECT command has two parameters, as shown in the command syntax:

  ```
  SELECT "appName" "dbName";
  ```

  If you enter only SELECT and press Enter, ESSCMD prompts you for the first parameter, the application name (*appName*). After you enter the application name and press Enter, ESSCMD prompts you for the database name (*dbName*).

- Type the commands and all parameters, then press Enter.

  Using SELECT as the example, you would type:

  ```
  SELECT "Sample" "Basic";
  ```

Whichever method you use, the interactive prompt now reflects the application and database names. For example, the following prompt tells you that the Sample application and Basic database are selected:

```
aspen:Sample:Basic:User[1]->
```

In this case, you can enter other commands without the application or database name parameters that it normally requires.

Canceling Operations

While ESSCMD is running, you can cancel an asynchronous operation, such as a calculation, export, or restructure operation, by pressing and holding the Esc key until ESSCMD responds.

CAUTION: Do not pause or suspend your system while Analytic Services is processing a command. Pausing the system may prevent Analytic Services from correctly completing the command.

Using Script and Batch Files for Batch Processing

If you use a series of commands frequently or you must enter many commands to complete a task, consider automating the task with a script or batch file. These files are useful for batch data loads and complex calculations.

For syntax conventions when working in batch processing mode, see "Understanding Syntax Guidelines" on page 1396.

- A *script file* contains ESSCMD commands. You can run a script file from the operating system command line or from within an operating system batch file, and the script file is processed by ESSCMD. By default, an ESSCMD script file has a `*.SCR` file extension. You can use a different extension.
- A *batch file* is an operating system file that calls multiple ESSCMD scripts, and can also include operating system commands. You can use a batch file to run multiple sessions of ESSCMD. You can run a batch file on Analytic Server from the operating system prompt; the file is processed by the operating system. On Windows NT, batch files have `*.bat` file extensions.

Note: On UNIX, a batch or script file is written as a shell script. A shell script usually has the file extension `.sh` (Bourne or Korn shell) or `.csh` (C shell).

When you run a script or batch file, ESSCMD executes the commands in order until it reaches the end of the file.

Existing script or batch files might be affected by changes to some commands. To ensure that your files work in this release, check on your use of changed or deleted commands. See the *Essbase Analytic Services Installation Guide* for information about new and changed commands.

Writing Script Files

ESSCMD script files automate an often-used or lengthy series of commands. Each script file must be a complete ESSCMD session, with login, application and database selection, logout, and termination commands.

➤ To define a script file:

1. Enter ESSCMD commands in any editor that saves data in ASCII text.
2. Save the file with the `.scr` ESSCMD script file extension.

   For example, the following script file, test`.scr`, was created in Notepad:

```
LOGIN "localhost" "User1" "password";
SELECT "Sample" "Basic";
GETDBSTATE
EXIT;
```

When run from the operating system command line, this script logs User1 into the Analytic Services localhost server, selects the Sample application and Basic database, gets database statistics, and quits the ESSCMD session.

Running Script Files

➤ To run script files in ESSCMD:

1. Enter the following command at the operating system prompt:

   `ESSCMD` *`scriptFileName`*`.scr`

2. Replace *scriptFileName* with the name of the script file. For example, type the following if the script file is in the current directory:

   `ESSCMD TEST.SCR`

3. If the script file is in another directory, include the path. For example:

```
ESSCMD C:\WORK\SCRIPTS\TEST.SCR (an absolute path on
Windows NT)
```

or

```
ESSCMD..\SCRIPTS\TEST.SCR (a relative path on Windows NT)
```

Handling Command Errors in Script Files

ESSCMD error-handling features provide error checking and handling for your script files. You can write error-handling commands into your script file to check for errors and, if necessary, branch to an appropriate error-handling response.

After each ESSCMD command is executed, a number is stored in an internal buffer. If the command executes successfully, `0` is returned to the buffer; If the command is unsuccessful, the error number is stored in the buffer; this is called *non-zero status*.

For error checking within an ESSCMD script file, ESSCMD provides the following error-handling commands:

- IFERROR checks the previously executed command for a non-zero return status (failure to execute). If the status is not zero, processing skips all subsequent commands and jumps to a user-specified point in the file, where it resumes. The script file can branch to an error-handling routine or the end of the file.
- RESETSTATUS reverts all saved status values to 0 (zero) in preparation for more status checking.
- GOTO forces unconditional branching to a user-specified point in the file, whether or not an error occurred.

In this `load.scr` example file, if the LOADDATA command does not execute successfully, ESSCMD branches to the end of the file to avoid attempting to calculate and run a report script on the empty database.

```
LOGIN "localhost" "User1" "password" "Sample" "Basic";
LOADDATA 2 "calcdat";
IFERROR "Error";
CALC "Calc All;";
IFERROR "Error";
RUNREPT 2 "Myreport";
```

```
IFERROR "Error";
[possible other commands]
EXIT;

:Error

EXIT;
```

Note: You can use the OUTPUT command to log errors to a text file.

For the syntax and usage of ESSCMD error commands, see the online *Technical Reference*, located in the *ARBORPATH*/docs directory.

Reviewing Sample Script Files

The following script files demonstrate common Analytic Services batch operations. All samples are based on the Sample Basic database that comes with your Analytic Services program. The scripts for these examples are available in your *ARBORPATH*\app\sample\basic directory. On UNIX systems, the examples are available from /home/hyperion/essbase/app/Sample/Basic.

Sample Script: Importing and Calculating Data

Suppose you need a file that executes the following actions:

- Logs on to Analytic Server.
- Selects an application and database.
- Prevents other users from logging on and making changes to the database.
- Imports data from a text file.
- Calculates the database.
- Enables logins again.
- Exits ESSCMD.

The following script file does the job:

```
LOGIN "Poplar" "TomT" "Password";
SELECT "Sample" "Basic";
DISABLELOGIN;
IMPORT 2 "ACTUALS" 4 "Y" 2 "ACTUAL" "N";
CALCDEFAULT;
ENABLELOGIN;
EXIT;
```

On Windows, this script file, `sample1.src`, is available in the `\ARBORPATH\app\Sample\Basic` directory. On UNIX platforms, `sample1.scr` is in the `/ARBORPATH/app/Sample/Basic` directory.

Sample Script: Building Dimensions and Importing and Calculating Data from a SQL Source

Suppose you need a script file that executes the following actions:

- Logs on to Analytic Server.
- Selects an application and database.
- Prevents other users from logging on and making changes to the database.
- Updates the outline from an SQL data source.
- Imports data from SQL.
- Calculates the database.
- Enables logins again.
- Exits ESSCMD.

The following script file does the job:

```
LOGIN "Poplar" "TomT" "Password";
SELECT "Sample" "Basic";
DISABLELOGIN;
BUILDDIM 2 "PRODRUL" 4 "PRODTBL" 4 "PROD.ERR";
IMPORT 4 "TOMT" "PASSWORD" 2 "ACTUAL" "N";
CALCDEFAULT;
EXIT;
```

On Windows, this script file, `sample2.scr`, is available in the `\ARBORPATH\app\Sample\Basic` directory. On UNIX systems, `sample2.scr` is in the `/ARBORPATH/app/Sample/Basic` directory.

Sample Script: Scheduling Report Printing

Suppose you need a file that executes the following actions:

- Logs on to Analytic Server
- Selects an application and database
- Assigns reports that output to files for later printing
- Exits ESSCMD

The following script file does the job:

```
LOGIN "Poplar" "TomT" "Password";
SELECT "Sample" "Basic";
RUNREPT 2 "REP1" "REP1.OUT";
RUNREPT 2 "REP2" "REP2.OUT";
RUNREPT 2 "REP3" "REP3.OUT";
EXIT;
```

On Windows, this script file, `sample3.scr`, is available in the `\ARBORPATH\App\Sample\Basic` directory. On UNIX platforms, `sample3.scr` is in the `/ARBORPATH/app/Sample/Basic` directory.

Writing Batch Files

You can write a batch file that runs one or more ESSCMD scripts, and includes operating system commands. See your operating system instructions to learn the syntax for writing batch files.

For examples of batch files, see “Reviewing Sample Script Files” on page 1406.

Handling Command Errors in Batch Files

For the operating system batch file, you can use ESSCMD command return values to control the flow of scripts that the batch file executes.

An ESSCMD program returns an integer value upon exiting. This value represents the status of the last executed command, usually whether the command succeeded or failed. You can set up your batch file to test for this value, and if the test fails, branch to an error-handling response. This process is similar to creating a script file. For information about handling errors in script files, see "Handling Command Errors in Script Files" on page 1405.

For example, a batch file could contain three scripts: an ESSCMD batch file that loads data, a calculation script that performs calculations on the data, and a report script that reports on the results of the calculation. If the load batch file fails, the calculations and reporting also fail. In this case, it would be best to stop the batch file after the failure of the load file and correct the error that caused the failure before going on. If your batch file tests for the return value of the load process, and this return value indicates failure, the batch file can jump to the end of the file and stop or execute some other error-handling procedure, rather than attempting to calculate data that did not load.

The following example shows an NT operating system batch file and the contents of one of the ESSCMD scripts it runs, `load.scr`. Because error-checking requirements vary, the syntax in this example may not correspond to that of your operating system. See your operating system documentation for error checking in batch files.

An operating system batch file could contain commands like this:

```
ESSCMD LOAD.SCR
If not %errorlevel%==0 goto Error
ESSCMD CALC.SCR
If not %errorlevel%==0 goto Error
ESSCMD REPORT.SCR
If not %errorlevel%==0 goto Error
Echo All operations completed successfully
EXIT

:Error
Echo There was a problem running the script
```

Glossary

! *See* bang character (!).

#MISSING. *See* missing data (#MISSING).

accounts dimension. A dimension type that makes accounting intelligence available. Only one dimension can be tagged as Accounts. An accounts dimension is not necessary.

Administration Server. A Java middle-tier server that handles communication between the Administration Services Console and Essbase Analytic Servers. It authenticates users as they connect to Analytic Servers.

Administration Services. An Essbase product used to manage the Essbase environment from a single point of access. It consists of a Java client console and middle-tier server that communicate directly with Essbase Analytic Servers.

Administration Services Console. A Java graphical user interface that communicates with Essbase Analytic Servers by way of the Essbase Administration Server. The console enables administrators to manage Analytic Servers from a single, graphical representation of the Essbase environment.

administrator. An individual who installs and maintains the Essbase system, including setting up user accounts and security. *See also* database administrator (DBA) and system administrator.

agent. A process on the server that starts and stops applications and databases, manages connections from users, and handles user-access security. The agent is referred to as ESSBASE.EXE.

aggregate. *See* consolidate.

aggregate cell. In aggregate storage databases, cells with values that are rolled up from lower-level cell values and potentially stored in aggregations. Input (level 0) cells and accounts dimension cells are not aggregate cells.

aggregate storage database. The Essbase database storage model supporting large-scale, sparsely distributed data which is categorized into many, potentially large dimensions. Selected data values are aggregated and stored, typically with improvements in aggregation time.

aggregate view. A collection of aggregate cells based on the levels of the members within each dimension. For example, a single aggregate view in a five-dimension outline can be represented as 0,0,1,0,2. The numbers are listed in the order of the dimensions in the outline. Each number represents the member level in the dimension. If the dimensions are Measures, Year, Months, Products, and Geography, the aggregate view 0,0,1,0,2 includes aggregate cells for intersections including level 0 members of Measures, Year, and Products; level 1 members of Months; and level 2 members of Geography. Aggregate views are not defined for accounts dimension levels above level 0.

aggregation. The process of rolling up and storing values in an aggregate storage database; the stored result of the aggregation process.

alias. An alternative name for a dimension, member, or description.

alias table. A database table that stores aliases for the dimensions or members.

Analytic Server. The server component of Analytic Services.

Analytic Server log. A record of actions performed by the Analytic Server (agent).

Analytic Services. The product that provides OLAP capabilities in the Hyperion platform. An instance of Analytic Services is an Analytic Server.

ancestor. A branch member that has members below it. For example, in a dimension that includes years, quarters, and months, the members Qtr2 and 2001 are ancestors of the member April.

application. A management structure containing one or more Essbase Analytic Services databases and the related files that control many system variables, such as memory allocation and autoload parameters.

application designer. An individual who designs, creates, and maintains Essbase Analytic Services applications and databases.

application log. A record of user actions performed on an application.

application programming interface (API). A library of functions that can be used in a custom program. Provides programmatic access to the data or services of an application.

area. A predefined set of members and values that makes up a partition.

arithmetic data load. A data load that performs operations on values in the database, such as adding 10 to each value.

asymmetric report. A report characterized by groups of members that differ by at least one member across the groups. There can be a difference in the number of members or the names of members under each heading in the report. For example, a report based on Sample Basic can have three members grouped under “East” and two members grouped under “West.”

attribute. A classification of a member in a dimension. For example, a Product dimension can have several attributes, such as Size and Flavor. A specific member of the Product dimension can have the Size attribute, 8, and the Flavor attribute, Cola.

attribute association. A relationship in a database outline whereby a member in an attribute dimension describes a characteristic of a member of its base dimension. For example, if product 100-10 has a grape flavor, the product 100-10 has the Flavor attribute association of grape. Thus, the 100-10 member of the Product dimension is associated with the Grape member of the Flavor attribute dimension.

Attribute Calculations dimension. A system-defined dimension that performs the following calculation operations on groups of members: Sum, Count, Avg, Min, and Max. This dimension is calculated dynamically and is not visible in the database outline. For example, by using the Avg member, you can calculate the average sales value for Red products in New York in January.

attribute dimension. A type of dimension that enables analysis based on the attributes or qualities of the members of its base dimension.

attribute reporting. A process of defining reports that is based on the attributes of the base members in the database outline.

attribute type. A text, numeric, Boolean, or date type that enables different functions for grouping, selecting, or calculating data. For example, because the Ounces attribute dimension has the type numeric, the number of ounces specified as the attribute of each product can be used to calculate the profit per ounce for that product.

axis. In MaxL DML, a specification determining the layout of query results from a database. For example, for a data query in Sample Basic, an axis can define columns for values for Qtr1, Qtr2, Qtr3, and Qtr4. Row data would be retrieved with totals in the following hierarchy: Market, Product. A different arrangement of data would require a different axis definition. For example, in the MaxL DML query "SELECT {Jan} ON COLUMNS FROM Sample.Basic", the axis specification is "{Jan} ON COLUMNS" (quotation marks are used for clarity but are not part of the syntax).

bang character (!). A character that terminates a series of report commands and requests information from the database. A report script must be terminated with a bang character; several bang characters can be used within a report script.

base currency. The currency in which daily business transactions are performed.

base dimension. A standard dimension that is associated with one or more attribute dimensions. For example, assuming products have flavors, the Product dimension is the base dimension for the Flavors attribute dimension.

batch calculation. Any calculation on a database that is done in batch; for example, a calculation script or a full database calculation. Dynamic calculations are not considered to be batch calculations.

batch file. An operating system file that can call multiple ESSCMD scripts and run multiple sessions of ESSCMD. On Windows-based systems, batch files have .BAT file extensions. On UNIX, batch files are written as a shell script.

batch processing mode. A method of using ESSCMD to write a batch or script file that can be used to automate routine server maintenance and diagnostic tasks. ESSCMD script files can execute multiple commands and can be run from the operating system command line or from within operating system batch files. Batch files can be used to call multiple ESSCMD scripts or run multiple instances of ESSCMD.

block. The primary storage unit within Essbase Analytic Services. A block is a multidimensional array representing the cells of all dense dimensions.

block storage database. The Essbase database storage model categorizing and storing data based on the sparsity of data values defined in sparse dimensions. Data values are stored in blocks which exist only for sparse dimension members for which there are values.

build method. A method used to modify database outlines. Choice of a build method is based on the format of data in data source files.

cache. A buffer in memory that holds data temporarily.

cache memory locking. An Essbase Analytic Services database setting that, when enabled, locks the memory used for the index cache, data file cache, and data cache into physical memory, potentially improving database performance. This setting is disabled by default.

calculated member in MaxL DML. In MaxL DML, a member designed for analytical purposes and defined in the optional WITH section of a MaxL DML query.

calculation. The process of aggregating or of running a calculation script on a database.

calculation script. A set of commands that define how a database is consolidated or aggregated. A calculation script may also contain commands that specify allocation and other calculation rules separate from the consolidation process.

cascade. The process of creating multiple reports for a subset of member values.

CDF. *See* custom-defined function (CDF).

CDM. *See* custom-defined macro (CDM).

cell. A unit of data representing the intersection of dimensions in a multidimensional database; the intersection of a row and a column in a worksheet.

cell note. A text annotation of up to 599 bytes for a cell in an Essbase Analytic Services database. Cell notes are a type of linked reporting object.

change log. *See* outline change log.

child. A member that has a parent above it in the database outline.

clean block. A data block marked as clean. A data block is clean if the database is fully calculated, if a calculation script calculates all dimensions at once, or if the SET CLEARUPDATESTATUS command is used in a calculation script.

client. (1) A client interface, such as the Essbase Spreadsheet Add-in software, a custom API program, or Essbase Administration Services Console. (2) A workstation that is connected to a server through a local area network.

code page. A mapping of bit combinations to a set of text characters. Different code pages support different sets of characters. Each computer contains a code page setting for the character set requirements of the language of the computer user. In the context of this document, code pages map characters to bit combinations for non-Unicode encodings. *See also* encoding.

column. A vertical display of information in a grid or table. A column can contain data from a single field, derived data from a calculation, or textual information. The terms column and field are sometimes used interchangeably. *Contrast with* row.

column heading. A part of a report that lists members across a page. When columns are defined that report on data from more than one dimension, nested column headings are produced. A member that is listed in a column heading is an attribute of all data values in its column.

committed access. An Essbase Analytic Services Kernel Isolation Level setting that affects how Analytic Services handles transactions. Under committed access, concurrent transactions hold long-term write locks and yield predictable results.

consolidate. The process of gathering data from dependent entities and rolling up the data up to parent entities. For example, if the dimension Year consists of the members Qtr1, Qtr2, Qtr3, and Qtr4, its consolidation is Year. The term roll-up also describes the consolidation process.

crosstab reporting. A type of reporting that categorizes and summarizes data in a table format. The cells within the table contain summaries of the data that fit within the intersecting categories. For example, a crosstab report of product sales information could show size attributes, such as Small and Large, as column headings and color attributes, such as Blue and Yellow, as row headings. The cell in the table where Large and Blue intersect could contain the total sales of all Blue products that are sized Large.

currency. The monetary unit of measure associated with a balance or transaction.

currency conversion. A process that converts currency values in a database from one currency into another currency. For example, to convert one U. S. dollar into the euro, the exchange rate (for example, 0.923702) is multiplied with the dollar (1* 0.923702). After conversion, the euro amount is .92.

currency partition. A dimension type that separates local currency members from a base currency, as defined in an application. A currency partition identifies currency types, such as Actual, Budget, and Forecast.

currency symbol. A character that represents a currency. For example, the currency symbol for the U. S. dollar is $ and the currency symbol for the British pound is £.

custom-defined function (CDF). Essbase Analytic Services calculation functions that are developed in the Java programming language and added to the standard Analytic Services calculation scripting language by means of MaxL. *See also* custom-defined macro (CDM).

custom-defined macro (CDM). Essbase Analytic Services macros that are written with Analytic Services calculator functions and special macro functions. Custom-defined macros use an internal Analytic Services macro language that enables the combination of calculation functions and they operate on multiple input parameters. *See also* custom-defined function (CDF).

cycle through. To perform multiple passes through a database while calculating it.

data block. *See* block.

data cache. A buffer in memory that holds uncompressed data blocks.

data cell. *See* cell.

data file. A file containing data blocks; Essbase Analytic Services generates the data file during a data load and stores it on a disk.

data file cache. A buffer in memory that holds compressed data (.PAG) files.

data load. The process of populating an Essbase Analytic Services database with data. Loading data establishes actual values for the cells defined by the structural outline of the database.

data load rules. A set of criteria or rules that Analytic Services uses to determine how to load data from a text-based file, a spreadsheet, or a relational data set into an Analytic Services database.

data mining. The process of searching through an Analytic Services database for hidden relationships and patterns in a large amount of data.

data security. Security set at the data level to control which data users have access.

data source. External data, such as a text file, spreadsheet file, relational database, or data warehouse that will be loaded into an Analytic Services database.

data value. *See* cell.

database. A repository of data within Essbase Analytic Services that contains a multidimensional data storage array. Each database consists of a storage structure definition (a database outline), data, security definitions, and optional calculation scripts, report scripts, and data loading scripts.

database administrator (DBA). An individual who administers database servers, such as Essbase Analytic Server, and who may also design, maintain, and create databases.

database designer. In Essbase Analytic Services, the highest type of access that can be assigned per database. This type of access allows complete calculate and update access as well as the ability to change the database design, export data, query against the database, and run report and calculation scripts.

DBA. *See also* database administrator (DBA).

dense dimension. In block storage databases, a dimension with a high probability that data exists for every combination of dimension members.

descendant. Any member below a parent in the database outline. For example, in a dimension that includes years, quarters, and months, the members Qtr2 and April are descendants of the member Year.

dimension. A data category that is used to organize business data for retrieval and preservation of values. Each dimension usually contains a hierarchy of related members grouped within it. For example, a Year dimension often includes members for each time period, such as quarters and months. Other common business dimensions may be measures, natural accounts, products, and markets.

dimension build. The process of adding new dimensions and members (without data) to a Essbase Analytic Services outline. *Contrast with* data load.

dimension build rules. Specifications, similar to data load rules, that Essbase Analytic Services uses to modify an outline. The modification is based on data in an external data source file.

dimension type. A dimension property that enables the use of predefined functionality. Dimensions that are tagged as Time have a predefined calendar functionality.

dimensionality. In MaxL DML, the represented dimensions (and the order in which they are represented) in a set. For example, the following set consists of two tuples of the same dimensionality because they both reflect the dimensions (Region, Year): { (West, Feb), (East, Mar) }

dirty block. A data block containing cells that have been changed since the last calculation. Upper level blocks are marked as dirty if their child blocks are dirty (that is, they have been updated).

disabled username. A inactive username, meaning that the user is not able to log on to Analytic Server.

drill down. The process of retrieving progressively detailed data relative to a selected dimension by expanding a parent member to reveal its children. For example, drilling down can reveal the hierarchical relationships between year and quarters or between quarter and months.

dynamic reference. A pointer in the rules file to header records in a data source. Header records define data load or dimension build criteria for the fields in a data source.

Dynamic Time Series. A process that is used to perform dynamic period-to-date reporting for all values associated with a query.

encoding. A method for mapping bit combinations to text characters for creating, storing, and displaying character text. Each encoding has a name; for example, UTF-8. Within a specific encoding, each character maps to a specific bit combination; for example, in UTF-8, uppercase A maps to HEX41. *See also* code page, UTF-8 (Unicode Transformation Format, 8-bit encoding format), and locale.

Enterprise View. An Administration Services feature that enables viewing and managing of the Essbase environment from a graphical tree view. From Enterprise View, you can operate directly on Analytic Services objects.

Essbase kernel. A layer of the Essbase Analytic Server that provides the foundation for a variety of functionality, including data loading, calculations, spreadsheet lock&send, partitioning, and restructuring. The Essbase kernel reads, caches, and writes data; it manages transactions; and it enforces transaction semantics to ensure data consistency and data integrity.

essbase.cfg. The name of an optional configuration file for Essbase Analytic Services. Administrators may enter parameters and values in this file to customize Analytic Server functionality. Some of the configuration settings may also be used with Analytic Services clients to override the Analytic Server settings.

EssCell. The Essbase Analytic Services cell retrieve function. An EssCell function is entered into a cell in Essbase Spreadsheet Add-in to retrieve a single database value that represents an intersection of specific database members.

ESSCMD. A command-line interface that is used to perform Analytic Services operations interactively or through a batch file.

ESSCMD script file. A text file that contains ESSCMD commands, executed in order to the end of the file. A script file can be run from the operating system command line or from within an operating system batch file. The default extension is .SCR.

ESSLANG. The Analytic Services environment variable that defines the encoding that Analytic Server uses to interpret text characters. *See also* encoding.

essmsh. *See* MaxL Shell.

external authentication. The practice of storing user logon credentials in a corporate authentication repository (such as a Lightweight Directory Access Protocol [LDAP] directory) as an alternative to maintaining users and groups that are native to each Hyperion product.

extraction command. A type of reporting command that handles the selection, orientation, grouping, and ordering of raw data extracted from a database. These commands begin with the less than (<) character.

field. A value or item in a data source file that will be loaded into an Essbase Analytic Services database.

file delimiter. One or more characters, such as commas or tabs, that separate fields in a data source.

filter. A method for controlling access to values and metadata in Essbase Analytic Services databases. A filter enables the placing of access restrictions on specific data and metadata for specific users.

formula. A combination of operators and calculation functions, as well as dimension names, member names, and numeric constants. Formulas are used to perform specific calculations on members of a database.

free-form reporting. A method of creating reports in which you type members of dimensions or report script commands in a worksheet. Free-form reporting is available in both Advanced Interpretation mode and Free-Form mode.

function. A predefined routine that returns a value, a range of values, a Boolean value, or one or more database members.

generation. A layer in a hierarchical tree structure that defines member relationships in a database. For example, generations are ordered incrementally from the top member of the dimension (generation 1) down to the child members.

generation name. A unique name that describes a generation.

global report command. A command that is executed when it occurs in the report script file and that stays in effect until the end of the report file or until another global command replaces it.

grouping. A set of members that is selected by a filtering process and that may be treated as a separate aggregate group. This group behaves very much like a parent to all of its specific members, and it supports full calculation logic, including additive and non-additive calculations. For example, you can use the attribute Small to view and work with all members with the attribute Small.

header record. One or more records at the top of a data source. Header records describe the contents of the data source.

hierarchy. A set of multidimensional relationships in an outline, often created in a tree formation. For example, parents, children, and generations represent a hierarchy.

Hybrid Analysis. The integration of a relational database with an Essbase Analytic Services multidimensional database so that lower-level data remains in the relational database and is mapped to summary-level data residing in the Analytic Services database. Hybrid Analysis enables Analytic Services to take advantage of the mass scalability of a relational database while maintaining a multidimensional view of the data in a live environment.

index. (1) A method that Essbase Analytic Services uses to retrieve data. The retrieval is based on the combinations of sparse dimensions. (2) The index file.

index cache. A buffer in memory that holds index pages.

index entry. A pointer to an intersection of sparse dimensions. Each index entry points to a data block on disk and locates a particular cell within the block by means of an offset.

index file. A file that Essbase Analytic Services uses to store data retrieval information. It resides on disk and contains index pages.

index page. A subdivision of an index file containing entries that point to data blocks.

input block. A type of data block that has at least one loaded data value.

input data. Any data that is loaded from a data source and is not generated by calculating the database.

intelligent calculation. A calculation method that tracks which data blocks have been updated since the last calculation.

interactive mode. A method of using ESSCMD by entering commands in the ESSCMD window and responding to prompts if necessary

interdimensional irrelevance. A situation in which a specific dimension does not intersect with other dimensions. The data is not irrelevant, but because the data in the specific dimension cannot be accessed from the other dimensions, those other dimensions are not relevant to the specific dimension.

isolation level. An Essbase Analytic Services kernel setting that determines the lock and commit behavior of database operations. Choices are committed access and uncommitted access.

latest. A key word that is used within Essbase Spreadsheet Add-in or within Report Writer to extract data values based on the member defined as the latest period of time.

layer. A generic way to refer to either a generation or a level. Refers to the location of members within the hierarchical tree structure of relationships in a database. That location is not specifically identified using numbered steps counted from the bottom up (as with levels) or from the top down (as with generations). For example, in the Sample Basic database, Qtr1 and Qtr4 are in the same layer.This means that Qtr1 and Qtr4 are also in the same generation. However, in a different database with a ragged hierarchy, Qtr1 and Qtr4 might not necessarily be in the same level simply because they are in the same generation.

leaf member. A member that has no children.

level. A layer within the hierarchical tree structure of member relationships in a database. The levels are numbered incrementally from leaf members (level 0) towards the top member of the dimension.

level 0 block. A data block that is created for sparse member combinations when all of the members of the sparse combination are level 0 members.

level 0 member. *See* leaf member.

level name. A unique name that describes a level.

linked object. A term that encompasses linked partitions and linked reporting objects.

linked partition. A form of shared partition that provides the ability to use a data cell to link together two different databases. When a user clicks a linked cell in a worksheet, for example, Essbase Analytic Services opens a new sheet displaying the dimensions in the second database. The user can then drill down into the available dimensions in the second database.

linked reporting object (LRO). An external file that is linked to a data cell in an Essbase Analytic Services database. Linked reporting objects (LROs) can be cell notes, URLs, or files that contain text, audio, video, or pictures.

locale. A computer setting that identifies the local language and cultural conventions such as the formatting of currency and dates, sort order of the data, and the character set encoding to be used on the computer. Analytic Services uses only the encoding portion of the locale. *See also* encoding and ESSLANG.

locale header record. An additional text record, at the beginning of some non-Unicode-encoded text files such as scripts, that identifies the encoding locale.

location alias. A descriptor that identifies a data source. The location alias specifies a server, application, database, username, and password. Location aliases are set by the database administrator at the database level using Application Manager, ESSCMD, or the API.

log. A system-maintained record of transactional data resulting from actions and commands.

Log Analyzer. An Administration Services feature that enables filtering, searching, and analysis of Essbase logs.

log delimiter. A character inserted between fields of a log file to allow a program to parse and manipulate log file information.

log file. A system-maintained file that records transactional data resulting from actions and commands. For example, an application log file records user actions that are performed on that application; a client log file records client messages, actions, and errors.

LRO. *See* linked reporting object (LRO).

mathematical operator. A symbol that defines how data is calculated. A mathematical operator can be any of the standard mathematical or Boolean operators; for example, +, -, *, /, and %. Mathematical operators are used in formulas and outlines.

MaxL. The multidimensional database access language for Essbase Analytic Services, consisting of a data definition language (MaxL DDL) and a data manipulation language (MaxL DML). MaxL DDL statements make possible the performance of batch or interactive system-administrative tasks on the Analytic Services system. MaxL DML statements enable the performance of data querying and extraction. *See also* MaxL Shell.

MaxL Perl Module. A Perl module (essbase.pm) that is part of the MaxL DDL component of Essbase Analytic Services. The essbase.pm module can be added to the Perl package to provide access to Analytic Services databases from Perl programs.

MaxL Script Editor. A script-development environment that is part of the Essbase Administration Console interface. The MaxL Script Editor is an integrated alternative to using a text editor and the MaxL Shell for creating, opening, editing, and running MaxL scripts for Essbase Analytic Services system administration.

MaxL Shell. An interface for passing MaxL statements to Essbase Analytic Server. The MaxLShell executable file, located in the bin directory for Essbase, is named essmsh (UNIX) or essmsh.exe (Windows).

member. A discrete component within a dimension. For example, a time dimension might include such members as Jan, Feb, and Qtr1.

member filtering (member selection). The process of selecting specific members that will be used in a query. Selection criteria can be applied, such as generation names, level names, pattern match, attributes, and UDAs.

member load. In Essbase Integration Services, the process of adding new dimensions and members (without data) to an Analytic Services outline. *Contrast with* data load.

member selection. *See* member filtering (member selection).

member selection report command. A type of Report Writer command that selects ranges of members based on database outline relationships, such as sibling, generation, and level.

member-specific report command. A type of Report Writer formatting command that is executed as it is encountered in a report script. The command affects only the member to which it is associated and executes the format command before it processes the member.

metadata sampling. The process of retrieving a portion of the members of a selected dimension when performing a drill-down operation. This portion of members is defined by the spreadsheet end user as a percentage of the members to retrieve during drill down. By drilling down to a portion of the members instead of all members in a dimension, retrieval is rapid and you can perform analysis with a focus on trends.

metadata security. Security set at the member level to control users from accessing certain members in an outline.

metaoutline. In Essbase Integration Services, a template containing the structure and rules for creating an Essbase Analytic Services outline from an OLAP model.

Minimum Database Access. An option group that controls the default security to all the databases of an application, using access settings (such as Read or None) that are applied globally to the application.

missing data (#MISSING). A marker indicating that data in the labeled location does not exist, contains no value, or was never entered or loaded. For example, missing data exists when an account contains data for a previous or a future period but not for the current period.

multidimensional database. A method of organizing, storing, and referencing data through three or more dimensions. An individual value is the intersection of a point for a set of dimensions.

multithreading. Within a single program, concurrent handling of multiple, separately executable sequences of program instructions.

named set. In MaxL DML, a set with its logic defined in the optional WITH section of a MaxL DML query. The named set can be referenced multiple times in the query.

nested column headings. A column heading format for report columns that displays data from more than one dimension. For example, in the Sample Basic database, a column heading that contains both Year and Scenario members is a nested column. This is scripted as follows: <COLUMN (Year, Scenario). The nested column heading shows Q1 (from the Year dimension) in the top line of the heading, qualified by Actual and Budget (from the Scenario dimension) in the bottom line of the heading.

numeric attribute range. A feature used to associate a base dimension member that has a discrete numeric value with an attribute that represents a range of values. For example, to classify customers by age, an Age Group attribute dimension can be defined that contains members for the following age ranges: 0-20, 21-40, 41-60, and 61-80. Each member of the Customer dimension can be associated with a particular Age Group range. Data can then be retrieved based on the age ranges rather than based on individual age values.

object. A program component that is related to an application or database. Objects can be outlines, rules files, calculation scripts, report scripts, or data sources. They are stored within the application or database subdirectory on the server or client machine.

OLAP. *See* online analytical processing (OLAP).

OLAP Metadata Catalog. In Essbase Integration Services, a relational database containing metadata describing the nature, source, location, and type of data that is pulled from the relational data source. Essbase Integration Server accesses the OLAP Metadata Catalog to generate the SQL statements and the information required to generate an Essbase Analytic Services database outline.

OLAP model. In Essbase Integration Services, a logical model (star schema) that is created from tables and columns in a relational database. The OLAP model is then used to generate the structure of a multidimensional database.

online analytical processing (OLAP). A multidimensional, multiple-user, client-server computing environment for users who need to analyze consolidated enterprise data. OLAP systems feature functionality such as drilling down, data pivoting, complex calculations, trend analyses, and modeling.

outline. The database structure of a multidimensional database, including all dimensions, members, tags, types, consolidations, and mathematical relationships. Data is stored in the database according to the structure defined in the outline.

outline change log. A record of changes made to an Essbase Analytic Services database outline.

outline synchronization. For partitioned databases, the process of propagating outline changes from the outline of one of the partitions to the outline of the other partition.

page file. *See* data file.

page heading. A type of report heading that lists members that are represented on the current page of the report. All data values on the page have the members in the page heading as a common attribute.

paging. A storage scheme that makes use of spare disk space to increase the available memory.

parallel calculation. An optional calculation setting. Essbase Analytic Services divides a calculation into tasks and calculates some of the tasks at the same time.

parallel data load. In Essbase Analytic Services, the concurrent execution of different stages of a single data load by multiple process threads.

parallel export. The ability to export Essbase Analytic Services data to multiple files. This may be faster than exporting to a single file, and it may resolve problems caused by a single data file becoming too large for the operating system to handle.

parent. A member that has an aggregated branch of children below it.

partition area. A subcube within a database. A partition is composed of one or more areas. These areas are composed of cells from a particular portion of the database. For replicated and transparent partitions, the number of cells within an area must be the same for both the data source and the data target to ensure that the two partitions have the same shape. If the data source area contains 18 cells, the data target area must also contain 18 cells to accommodate the number of values.

partitioning. The process of defining areas of data that are shared or linked between data models. Partitioning can affect the performance and scalability of Essbase Analytic Services applications.

Password Management. A group of options in the server settings that are used to limit a user's allowed number of login attempts, number of days of inactivity, and number of days using the same password.

pattern matching. The ability to match a value with any or all characters of an item that is entered as a criterion. A missing character may be represented by a wild card value such as a question mark (?) or an asterisk (*). For example, "Find all instances of apple" returns apple, but "Find all instances of apple*" returns apple, applesauce, applecranberry, and so on.

period. An interval within the time dimension.

permission. A level of access granted to users and groups for managing data or other users and groups.

persistence. The continuance or longevity of effect for any Essbase Analytic Services operation or setting. For example, an Analytic Services administrator may limit the persistence of username and password validity.

Personal Essbase. A version of the Analytic Server that is designed to run on one computer.

pivot. The ability to alter the perspective of retrieved data. When Essbase Analytic Services first retrieves a dimension, it expands data into rows. You can then pivot or rearrange the data to obtain a different viewpoint.

precalculation. The process of calculating the database prior to user retrieval.

preserve formulas. The process of keeping user-created formulas within a worksheet while retrieving new data.

property. A characteristic of a member, such as two-pass calculation or shared member. Properties affect how Essbase Analytic Services works with the data.

query governor. An Essbase Integration Server parameter or Essbase Analytic Server configuration setting controls the duration and size of the queries made to the data source.

record. In a database, a group of fields that make up one complete entry. For example, a record about a customer might contain fields for name, address, telephone number, and sales data.

redundant data. Duplicate data blocks that Essbase Analytic Services retains during transactions until Analytic Services commits the updated blocks.

replicated partition. A portion of a database, defined through Partition Manager, that is used to propagate an update to data that is mastered at one site to a copy of data that is stored at another site. Users are able to access the data as though it were part of their local database.

report. The formatted summary information that is returned from a database after a report script is run. One or more reports can be generated from a report script.

Report Extractor. An Essbase Analytic Services component that retrieves report data from the Analytic Services database when a report script is run.

report script. A text file containing Essbase Analytic Services Report Writer commands that generate one or more production reports. Report scripts can be run in batch mode, through the ESSCMD command-line interface, or through Essbase Administration Services. The report script is a text file that contains data retrieval, formatting, and output instructions.

Report Viewer. An Essbase Analytic Services component that displays the complete report after a report script is run. Saved reports typically show the file extension .RPT.

request. A query sent to Essbase Analytic Server by a user or by another process; for example, starting an application, or restructuring a database outline. Requests happen in the context of sessions. Only one request at a time can be processed in each session. A request can be terminated by another user with the appropriate permissions (for example, by an administrator). *See also* session.

restore. An operation to reload data and structural information after a database has been damaged or destroyed. The restore operation is typically performed after you shut down and restart the database.

restructure. An operation to regenerate or rebuild the database index and, in some cases, the data files.

root member. The highest member in a dimension branch.

row. A horizontal display of information in a grid or table. A row can contain data from a single field, derived data from a calculation, or textual information. The words row and record are sometimes used interchangeably. *Contrast with* column.

row heading. A report heading that lists members down a report page. The members are listed under their respective row names.

sampling. The process of selecting a representative portion of an entity for the purpose of determining the characteristics of that entity. *See also* metadata sampling.

scope. The area of data encompassed by any Essbase Analytic Services operation or setting; for example, the area of data affected by a security setting. Most commonly, scope refers to three levels of granularity, where higher levels encompass lower levels. From highest to lowest, these levels are as follows: the entire system (Essbase Analytic Server), applications on Analytic Server, or databases within Analytic Server applications. *See also* persistence.

security platform. A framework providing the ability for Hyperion applications to use external authentication and single sign-on.

serial calculation. The default calculation setting. Essbase Analytic Services divides a calculation pass into tasks and calculates one task at a time.

server. A multiple-user computer that accesses data values based on the intersection of dimension members.

server application. *See* server.

server console. For Essbase Analytic Services, the computer on which Agent commands are entered and messages from the Agent are shown. If you run Analytic Services in the foreground on either Windows or UNIX, you can enter Agent commands. On UNIX, a telnet session is used to access Analytic Services remotely. On Windows, Analytic Services can be accessed only from the server console.

server interruption. Any occurrence that stops the server, including an abnormal shutdown, a power outage, or a user pressing the Ctrl+C keys.

session. The time between login and logout for a user connected to Essbase Analytic Server.

set. In MaxL DML, a required syntax convention for referring to a collection of one or more tuples. For example, in the following MaxL DML query, SELECT { [100-10] } ON COLUMNS FROM Sample.Basic { [100-10] } is a set.

shared member. A member that shares storage space with another member of the same name. A storage property designates members as shared. The use of shared members prevents duplicate calculation of members that occur more than once in an Essbase Analytic Services outline.

sibling. A child member at the same generation as another child member and having the same immediate parent. For example, the members Florida and New York are both children of the member, East, and siblings of each other.

single sign-on. The ability of an externally-authenticated user to access multiple, linked Hyperion applications after logging on to the first application. The user can launch other applications from the first application (and from other linked Hyperion applications) without logging on again. The user's ID and password are already authenticated.

slicer. In MaxL DML, the section at the end of a query that begins with and includes the keyword WHERE.

sparse dimension. In block storage databases, a dimension with a low probability that data exists for every combination of dimension members.

Spreadsheet Add-in. Essbase Analytic Services software that works with a spreadsheet. Essbase Spreadsheet Add-in is an add-in module to spreadsheet software.

standard dimension. A dimension that is not an attribute dimension.

subset. A cross-section of data. Subsetting further defines members that meet specific conditions.

substitution variable. A variable that acts as a global placeholder for information that changes regularly. The variable and a corresponding string value is set; the value can then be changed at any time. Substitution variables can be used in calculation scripts, report scripts, Essbase Spreadsheet Add-in, Essbase API, and Hyperion Planning forms.

suppress rows. The option to exclude rows that contain missing values and to underscore characters from spreadsheet reports.

swapping. *See* paging.

symmetric multiprocessing (SMP). A server architecture that enables multiprocessing and multithreading. Essbase Analytic Server supports multiple threads over SMP servers automatically. Thus, performance is not significantly degraded when a large number of users connect to an single instance of Analytic Server simultaneously.

system administrator. A person who maintains the hardware, software, disk space distribution, and configurations for running software applications, such as Essbase Analytic Services.

TCP/IP. *See* Transmission Control Protocol/Internet Protocol (TCP/IP).

template. A predefined format that is designed to retrieve particular data on a regular basis and in a consistent format.

time series reporting. A process of reporting data based on a calendar date (for example, year, quarter, month, or week).

toolbar. A series of shortcut buttons providing quick access to commands. The toolbar is usually located directly below the menu bar. Not all windows display a toolbar.

Transmission Control Protocol/Internet Protocol (TCP/IP). A standard set of communication protocols that is adapted by many companies and institutions around the world and that links computers with different operating systems and internal architectures. TCP/IP utilities are used to exchange files, send mail, and store data to various computers that are connected to local and wide area networks.

transparent partition. A form of shared partition that provides the ability to access and manipulate remote data transparently as though it is part of a local database. The remote data is retrieved from the data source each time a request is made. Any updates made to the data are written back to the data source and become immediately accessible to both local data target users and transparent data source users.

triggers. An Essbase Analytic Services feature that enables efficient monitoring of data changes in a database. If data breaks rules that you specified, Analytic Services alerts the user or system administrator.

tuple. In MaxL DML, a required syntax convention for referring to a member or a member combination from any number of dimensions. For example, in the Sample Basic database, (Jan) is a tuple, and so is (Jan, Sales), and so is ([Jan], [Sales], [Cola], [Texas], [Actual]).

UDA. A user-defined attribute. A UDA is a term associated with members of an outline to describe a particular characteristic of the members. Users can specify UDAs within calculation scripts and reports so that they return lists of members that have the specified UDA associated with them. UDAs can be applied to dense as well as sparse dimensions.

unary operator. A group of mathematical indicators (+, -, *, /, %) that define how roll-ups take place on the database outline.

uncommitted access. An Essbase Analytic Services kernel setting that affects how Analytic Services handles transactions. Under uncommitted access, concurrent transactions hold short-term write locks and can yield unpredictable results.

Unicode. An approach to encoding character text such that thousands of text characters from hundreds of different languages are all supported within a single encoding form. *See also* encoding.

Unicode-mode Analytic Server. An Analytic Server with the setting enabled so that Unicode-mode applications can be created and non-Unicode-mode applications can be migrated to be Unicode-mode applications. *See also* Unicode-mode application.

Unicode-mode application. An Analytic Services application wherein character text is encoded in UTF-8, enabling users with their computers set up for different languages to share the application data.

Uniform Resource Locator (URL). An address for a resource located on the World Wide Web, such as a document, image, downloadable file, service, or electronic mailbox. URLs use a variety of naming schemes and access methods, such as HTTP, FTP, and Internet mail. An example of a URL is http://www.hyperion.com. A URL can also point to a file located on a local or network drive, such as file:///D:/essbase/docs/essdocs.htm.

URL. *See* Uniform Resource Locator (URL).

UTF-8 (Unicode Transformation Format, 8-bit encoding format). A Unicode encoding format that maps characters in a sequence of one to four bytes for each character. The first 128 characters of this encoding form use the same bit combinations as the common Latin1 encoding used for English characters. *See also* Unicode and encoding.

UTF-8 signature. An internal mark, at the beginning of a text file, that indicates that the file is encoded in UTF-8.

validation. A process of checking a rules file, report script, or partition definition against the outline to make sure that the object being checked is valid.

visual cue. A formatted style, such as a font or a color, that highlights specific types of data values. Data values may be dimension members; parent, child, or shared members; dynamic calculations; members containing a formula; read only data cells; read and write data cells; or linked objects.

WITH section. In MaxL DML, an optional section of the query used for creating re-usable logic to define sets or members. Sets or custom members can be defined once in the WITH section, and then referenced multiple times during a query.

workbook. An entire spreadsheet file with many worksheets.

write-back. The ability for a retrieval client, such as a spreadsheet, to update a database value.

Index

Symbols

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Numerics

A

B

C

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

D

E

F

G

H

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

I

J

K

L

M

N

O

P

Q

R

S

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

T

U

V

W

X

Y

Z