

# Oracle® Hyperion Financial Management Administrator's Guide



Release 11.2.x  
F13247-14  
March 2025

The Oracle logo, consisting of a solid red square with the word "ORACLE" in white, uppercase, sans-serif font centered within it.

ORACLE®

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

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

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

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

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

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

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

# Contents

## Documentation Accessibility

---

## Documentation Feedback

---

### 1 About Financial Management

---

Financial Management Features	1-1
EPM Workspace	1-1
Financial Management Dimensions	1-2
Scenario Dimension	1-2
Year Dimension	1-2
Period Dimension	1-2
Entity Dimension	1-2
Value Dimension	1-3
Account Dimension	1-3
Intercompany Dimension	1-3
View Dimension	1-3
Custom Dimensions	1-3
User-Defined Elements	1-4

### 2 Managing Applications

---

Application Administration	2-1
Creating Applications	2-2
Creating a New Application	2-2
Defining Frequencies	2-4
Editing Periods	2-5
Adding Sibling and Child Periods	2-6
Deleting Periods	2-7
Defining Features	2-7
Specifying Application Settings	2-7
Enabling or Disabling Application Modules	2-8
Creating Custom Dimensions	2-9

Saving Application Profiles	2-11
Creating an Application from a Profile File	2-11
Copying from an Application	2-11
Modifying Applications	2-12
Registering Applications	2-13
Opening Applications	2-13
Closing Applications	2-13
Changing Application Server Clusters	2-13
Viewing the Application List	2-14
Deleting Applications	2-14
Loading Application Elements	2-14
Extracting Application Elements	2-15
Duplicating Financial Management Applications	2-16
Importing Applications	2-17
Creating a Database Link for an Oracle Database	2-18
Creating a Linked Server for a SQL Server Database	2-18
Working with System Messages	2-20
Viewing System Messages	2-20
Deleting System Messages	2-20
System Message Detail Strings	2-21
Managing Application Access	2-21
Logging Out Users	2-22
Disabling and Enabling Connections	2-22
Viewing Application Connection Status	2-23
Managing System Users	2-23
Viewing Users	2-23
Logging Out Selected or All Users	2-24
Managing Servers and Applications	2-25
Enabling and Disabling Admin Mode	2-25
Synchronizing Servers	2-25
Auditing Tasks	2-25
Auditing Data	2-28
Monitoring Running Tasks	2-29
Viewing Running Tasks	2-30
Stopping Running Tasks	2-31
Refreshing Running Tasks	2-31
Scanning For and Clearing Invalid Records	2-32
Monitoring System Status Using HFM Insights	2-32
Accessing HFM Insights	2-32
HFM Insights Main Dashboard	2-33
Filtering Applications in HFM Insights	2-33

### 3 Managing Application Security

---

Application Security Considerations	3-1
Launching the Shared Services Console from Financial Management	3-2
Selecting Users and Groups for Assigning Security Classes	3-2
Setting Up Security Classes	3-3
Creating Security Classes	3-3
Deleting Security Classes	3-4
Selecting Security Classes	3-4
Assigning User Access to Security Classes	3-5
Setting Up Email Alerting	3-6
Running Security Reports	3-6
Loading Application Security	3-7
Clearing and Loading Security Information	3-9
Before Clearing Security Information	3-9
After Clearing Security Information	3-10
Extracting Application Security	3-11

### 4 Managing Metadata

---

Defining Accounts	4-1
Account Type Behavior	4-5
Defining Dynamic Accounts	4-8
Defining Custom Members	4-8
Defining Entity Members	4-10
Defining Scenario Members	4-11
Defining Application Settings	4-14
Organization by Period	4-17
Defining Consolidation Methods	4-18
Using Consolidation Methods	4-19
Assigning Consolidation Methods Manually	4-20
Using POWN or POWNMIN Consolidation Methods	4-20
Defining Currencies	4-22
Defining Cell Text Labels	4-23
System-Generated Accounts	4-26
Consolidation Accounts	4-26
Ownership Accounts	4-27
Editing System-Generated Accounts	4-27
Setting Up Intercompany Partners	4-28
Editing System-Generated ICP Members	4-29

Editing System-Generated Value Members	4-29
Metadata Filtering Based on Security	4-30
Creating Metadata Files of the APP Format	4-31
File Format	4-32
Version	4-33
Application Settings	4-33
Currencies	4-35
Members	4-35
Account	4-36
Scenario	4-36
Entity	4-37
Custom	4-37
Value	4-38
Intercompany Partner	4-39
Consolidation Methods	4-40
Hierarchies	4-40
Account Hierarchies	4-41
Scenario Hierarchies	4-43
Entity Hierarchies	4-43
Custom Hierarchies	4-43
Dimensions Not Included in Metadata Files	4-44
Metadata Referential Integrity	4-44
Metadata Referential Integrity Checks	4-45
Metadata Log File Referential Integrity Errors	4-45
Using the Metadata Merge Utility	4-45
Loading Metadata	4-47
Viewing Metadata Load Changes	4-50
Extracting Metadata	4-51

## 5 Managing Member Lists

---

Creating Member List Files	5-1
EnumMemberLists	5-2
EnumMembersInList	5-3
Dynamic Member Lists	5-6
Dynamic POV Member List	5-7
Loading Member Lists	5-9
Extracting Member Lists	5-11
System Lists by Dimension	5-11

## 6 Managing Journals

---

Creating Journal Files	6-1
File Format Section	6-2
Version Section	6-2
Journal Group Section	6-2
Standard Section	6-3
Recurring Section	6-3
Header Section	6-3
Loading Journals	6-5
Extracting Journals	6-7

## 7 Managing Data Forms

---

Creating Data Forms in the Form Designer	7-1
Setting the Point of View	7-2
Specifying Form Details	7-3
Specifying On-Demand Rules for Data Forms	7-3
Specifying Form Row and Column Options	7-4
Specifying Form Headers	7-4
Using Data Form Scripts	7-5
AddMember	7-8
BackgroundPOV	7-9
Blank	7-11
Cn	7-11
Calc1	7-12
CalcByRow	7-12
Cell_Link	7-13
CellText	7-14
CustomHeader	7-14
CustomHeaderStyle	7-15
DynamicPOV	7-16
FormInputBoxLength	7-16
FormNumDecimals	7-16
FormRowHeight	7-17
FormScale	7-17
HeaderOption	7-18
HideInPov	7-19
Instructions	7-19
LineItemDetailSinglePeriod	7-20
Link	7-20
MaxCells	7-21

MaxColsForSparseRetrievalMethod	7-21
NoSuppress	7-22
NumDecimals	7-22
OnDemandRules	7-22
Override	7-22
POVOrder	7-24
PrintNumDataColsPerPage	7-24
PrintNumRowsPerPage	7-25
PrintRepeatHeadersonAllPages	7-25
Rn	7-26
ReadOnly	7-27
ReportDescription	7-27
ReportLabel	7-28
ReportSecurityClass	7-29
ReportType	7-29
RowHeaderPct	7-29
SCalc	7-30
Scale	7-31
SelectablePOVList	7-31
ShowDescriptions	7-32
ShowLabels	7-33
String	7-33
Style	7-33
SuppressColHeaderRepeats	7-35
SuppressInvalidCols	7-36
SuppressInvalidRows	7-36
SuppressNoDataCols	7-37
SuppressNoDataRows	7-37
SuppressRowHeaderRepeats	7-37
SuppressZeroCols	7-38
SuppressZeroRows	7-38
Using Relative Time Periods	7-38
Order of Precedence for Conflicting Attributes	7-39
Editing Data Forms	7-40
Loading Data Forms	7-40
Extracting Data Forms	7-41
Deleting Data Forms	7-42

## 8 Extracting Data to a Database

---

Configuring a Data Source Name (DSN)	8-1
Star Schemas	8-2



Star Schema Formats	8-3
Prefix Tables	8-4
Creating and Exporting Data to a Star Schema	8-4
Updating a Star Schema	8-6
Deleting a Star Schema	8-7
Creating a Star Schema Template	8-8
Deleting a Star Schema Template	8-8

## 9 Defining Reports

---

Defining Journal Report Scripts	9-1
Defining Intercompany Matching Report Scripts	9-2
Selecting Member Lists for Intercompany Matching Reports	9-3
Selecting Accounts for Intercompany Matching Reports	9-3
Specifying Decimal Places in Intercompany Matching Reports	9-3
Selecting Style Sheets for Intercompany Matching Reports	9-3
Specifying Currencies in Intercompany Matching Reports	9-3
Suppression Options for Intercompany Matching Reports	9-4
Intercompany Matching Report Script Keywords	9-5
ReportType	9-5
ReportDescription	9-5
StyleSheet	9-5
POV	9-5
Entity	9-6
Partner	9-6
AccountEntity and AccountPartner	9-7
SuppressIfMatch	9-9
SuppressReversedTransactions	9-10
SuppressDetails	9-10
MatchingTolerance	9-11
SuppressCustoms	9-11
ScaleFactor	9-12
Decimal	9-12
DisplayLabels	9-12
DisplayDescriptions	9-13
DisplayPlugElimAccts	9-13
GroupByCustom	9-13
Defining Intercompany Transaction Report Scripts	9-14

## 10 Managing Rules

---

Rule Types	10-1
Rule Considerations	10-4
Calculation Commands	10-4
Force Calculate	10-5
Calculation Rules with Calculation Commands	10-5
Calculation Rules with Consolidation Commands	10-6
Current Dimension Members	10-9
Account Expressions	10-10
Functions Automatically Clear Data	10-11
Error Messages	10-11
Rule Execution During Consolidation	10-12
Default Translation	10-13
Financial Management Objects	10-13
Using VBScript in Rules	10-14
VBScript Operators	10-14
VBScript Statements	10-15
VBScript Keywords	10-16
VBScript Functions	10-16
Date and Time Functions	10-16
Format Functions	10-17
Conversion Functions	10-17
Math Functions	10-17
Array Functions	10-17
String Functions	10-18
Other Functions	10-18
VBScript Objects	10-19
Commonly Used Rules	10-19
Reusing Data	10-20
Setting Accounts by Calculating Amounts	10-20
Conditional Rules	10-21
Testing for a Dimension Member	10-21
Testing for More Than One Member	10-22
Performing Different Actions for Different Members	10-23
Setting Opening Balances of All Accounts	10-23
Creating Rules Files	10-24
Loading Rules	10-26
Extracting Rules	10-27
Rules Profiling	10-28
Adding Profile Information to Rule Files	10-28
Starting and Stopping Rules Profiling	10-28

Viewing Rules Profile Results	10-29
Rules View	10-29
Entity View	10-30
Deleting Rules Profiles	10-31

## 11 Rule Functions

---

Functions Overview	11-1
ABSExp	11-8
AccountType	11-9
AccountTypeID	11-10
AddEntityToList	11-11
AddEntityToListUsingIDs	11-12
AddMemberToList	11-12
AddMemberToListUsingIDs	11-13
Alloc	11-13
AllowAdjFromChildren	11-15
AllowAdjs	11-16
ApplicationName	11-17
CalculateExchangeRate	11-18
CalculateRate	11-18
CalcStatus	11-19
CellTextUnitItem	11-21
Clear	11-23
Con	11-23
Consol1, Consol2, Consol3	11-25
ConsolidateYTD	11-25
ContainsCellText	11-26
ContainsCellTextWithLabel	11-27
Currency	11-28
CustomTop	11-29
DataUnitItem	11-30
Decimal	11-31
DefaultFreq	11-32
DefaultParent	11-33
DefaultTranslate	11-34
DefaultView	11-35
DefCurrency	11-36
Down	11-37
Dynamic	11-38
Exp	11-39
Dimension Intersection Considerations	11-40

Period and Year Keywords	11-41
Mathematical Calculations	11-42
Placing Other Functions Within Exp	11-42
Simultaneously Setting Several Accounts	11-42
Exp and Dimension Intersection Considerations	11-43
All Intersections Valid	11-44
Invalid Intersections	11-44
One Valid Member on the Right Side	11-45
GetCell	11-46
GetCellNoData	11-47
GetCellRealData	11-48
GetCellText	11-49
GetCellTextWithLabel	11-49
GetCellType	11-50
GetCustomLabelArray	11-51
GetItem	11-52
GetItemIDs2	11-53
GetItemIDs2ExtDim	11-54
GetNumItems	11-55
GetNumLID	11-55
GetRate	11-56
GetSubmissionGroup	11-57
GetSubmissionPhase	11-57
Holding	11-58
ICPTopMember	11-59
ICPWeight	11-60
IDFromMember	11-60
ImpactStatus	11-61
Input	11-62
IsAlmostEqual	11-63
IsBase	11-64
IsCalculated	11-66
IsChild	11-67
IsConsolidated	11-69
IsDescendant	11-70
IsFirst	11-72
IsICP	11-73
IsLast	11-74
IsTransCur	11-75
IsTransCurAdj	11-75
IsValidDest	11-76
IsZero	11-77

List	11-79
Member	11-80
MemberFromID	11-81
Method	11-82
NoInput	11-83
NoRound	11-83
NumBase	11-84
Number	11-86
NumChild	11-87
NumCustom	11-88
NumDescendant	11-89
NumPerInGen	11-91
NumPeriods	11-92
OpenCellTextUnit	11-93
OpenDataUnit	11-94
OpenDataUnitSorted	11-94
Owned	11-95
Owner	11-95
PCon	11-96
PEPU	11-96
PeriodNumber	11-97
PlugAcct	11-98
POwn	11-99
PVAForBalance	11-100
PVAForFlow	11-101
RateForBalance	11-101
RateForFlow	11-102
ReviewStatus	11-103
ReviewStatusUsingPhaseID	11-104
Round	11-104
Scale	11-105
SecurityAsPartner	11-107
SecurityClass	11-108
SetCellTextWithLabel	11-109
SetData	11-110
SetDataWithPOV	11-111
SubmissionGroup	11-111
SupportsProcessManagement	11-112
SupportsTran	11-113
SwitchSign	11-114
SwitchType	11-115
Trans	11-117

TransPeriodic	11-117
UD1...3	11-118
ValidationAccount	11-120
ValidationAccountEx	11-121
XBRLTags	11-121

## 12 Custom Functions

---

Management Reporting Functions	12-1
Average	12-2
Cumulative	12-5
Difference	12-8
DSO - Days Sales Outstanding	12-11
Opening	12-14
Rate	12-17
Business Rules Functions	12-23
Custom_Alloc	12-23
Increase_Decrease	12-27
Pro_Rata_Ratio	12-29
Spread	12-31
Units_Rates	12-34

## 13 Creating Rules Using Calculation Manager

---

Calculation Manager Security Roles	13-1
Working with Applications in Calculation Manager	13-1
Migrating Rules to Calculation Manager	13-2
VB Function Support in Function Selector	13-2
Special VB Script Functions for Financial Management	13-3

## 14 Managing Intercompany Transactions

---

Setting Up Intercompany Transactions	14-1
Opening Intercompany Periods	14-1
Setting Matching Tolerances	14-2
Setting Match/Validate Before Post Option	14-3
Closing Intercompany Periods	14-3
Managing Reason Codes	14-4
Adding Reason Codes	14-4
Editing Reason Codes	14-5
Deleting Reason Codes	14-5
Monitoring Intercompany Transactions	14-5

Locking and Unlocking Entities	14-7
Viewing the Intercompany Transactions Summary	14-8

## 15 Managing Process Management Submission Phases

---

Defining Submission Phases	15-1
Setting Up Submission Groups	15-2
Submission Group and Phase Examples	15-3
Assigning Submission Groups to Phases	15-5
Viewing Unassigned Submission Groups	15-5

## 16 Managing Email Alerting

---

Setting Up Process Management Alerting	16-1
Setting Up Intercompany Alerting	16-2

## A Configuration Settings

---

Available Configuration Settings	A-2
Changing Configuration Settings	A-8
Overriding Values	A-8
Changing the Settings Table Display	A-9
Searching for Settings	A-9
Viewing Effective Settings	A-10
Exporting Settings	A-10
Deleting Settings	A-10

## B Optimizing Performance

---

Performance Overview	B-1
Common EPM Installation Directory References	B-1
Financial Management Records and Subcubes	B-2
Tuning Recommendations for Financial Management	B-3
Diagnosing Performance Problems	B-3
Using Monitoring Tools	B-3
Using Remote Diagnostic Agent (RDA)	B-5
Using a Reference Application	B-5
Tuning Operating Systems Parameters	B-6
Tuning Windows Parameters	B-6
Tuning the Web Server	B-7
Tuning HFM Web	B-7
HFM Web Timeout Parameters	B-7

HFM Web Tuning Parameters	B-8
Tuning Financial Management Applications	B-10
Commonly Tuned Financial Management Settings	B-10
MaxNumDataRecordsInRAM	B-10
MinDataCacheSizeInMB	B-10
MaxDataCacheSizeInMB	B-11
MaxNumCubesInRAM	B-11
IcmSystemReportTranslationMode	B-11
NumConsolidationThreads	B-12
MaxNumConcurrentConsolidations	B-13
NumThreadsToUseWhenUpdatingCalcStatusSystemWasChanged	B-13
Financial Management Memory Settings for Larger Applications	B-13
Application-Specific Settings	B-14
Tuning Financial Management Application Servers	B-15
Financial Management Tuning Example	B-15
Application Database Maintenance	B-16
Basic Design Considerations	B-16
Tuning Oracle 11g Databases for Financial Management	B-17
Introduction	B-17
Common Performance Issues	B-18
Tuning Guidelines for Oracle 11g Databases	B-18
Oracle Initialization Parameters	B-18
CURSOR_SHARING	B-18
MEMORY_TARGET	B-19
MEMORY_MAX_TARGET	B-19
SGA_TARGET	B-20
SGA_MAX_SIZE	B-20
PGA_AGGREGATE_TARGET	B-20
LOG_BUFFER	B-20
OPTIMIZER_MODE	B-20
OPTIMIZER_INDEX_COST_ADJ	B-21
PROCESSES	B-21
SESSIONS	B-21
TRANSACTIONS	B-21
OPEN_CURSORS	B-21
SESSION_CACHED_CURSORS	B-21
TRACE_ENABLED	B-22
STATISTICS_LEVEL	B-22
TIMED_STATISTICS	B-22
TIMED_OS_STATISTICS	B-22
How to Determine Memory Settings for Oracle Database Release 11g	B-22
Total Memory Sizing (MEMORY_TARGET)	B-22



SGA Sizing (SGA_TARGET)	B-24
PGA Sizing (PGA_AGGREGATE_TARGET)	B-24
LOG_BUFFER Sizing	B-25
How to Calculate the Number of Processes for Oracle Database Release 11g	B-25
Other Considerations	B-26
Shared Server versus Dedicated Server	B-26
Online Redo Log Files Size	B-26
Tablespaces and Segments Fragmentations	B-26
Index Fragmentation	B-27
Disable the feature DEFERRED_SEGMENT_CREATION	B-27
Disable the feature DEFERRED_SEGMENT_CREATION	B-27
Regular Maintenance and Tuning Plans	B-28
Frequently Asked Questions	B-28

# Documentation Accessibility

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

## **Access to Oracle Support**

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

# Documentation Feedback

To provide feedback on this documentation, click the feedback button at the bottom of the page in any Oracle Help Center topic. You can also send email to [epmdoc\\_ww@oracle.com](mailto:epmdoc_ww@oracle.com).

# 1

## About Financial Management

### Related Topics

- [Financial Management Features](#)
- [EPM Workspace](#)
- [Financial Management Dimensions](#)
- [User-Defined Elements](#)

## Financial Management Features

Oracle Hyperion Financial Management provides these features:

- A unified view of enterprise financial information consolidates key performance and operating metrics from global sources in a scalable, Web-based application.
- "Fast virtual close" features trim days and weeks off your close cycle including using Web-based process management, Web-based intercompany reconciliations, journal adjustments and a consistent set of data and business measures.
- Powerful multidimensional analysis helps identify and report on key financial and business trends, new sources of profitability and cash flow at corporate, cost center, product, brand, customer, and channel levels.
- Flexible "what if" scenario management feature dynamically consolidates and reports actual results, financial budgets, forecasts and plans, producing new statements as assumptions and facts change.
- High-volume, preformatted reports deliver timely, accurate financial information for internal management and external regulatory and government bodies from the same application.
- Prepackaged features are deployed out-of-the-box, quickly and cost-effectively, including features such as world-class allocations, multicurrency translations, and robust data integration with legacy applications, ERP, and CRM systems.
- Customizable and extensible application solves your issues quickly and cost-effectively, using industry standard tools.
- Architected for the Web so users can easily and securely access global financial information from any location, using a standard Web browser. Relational data storage ensures mission critical data is available to users 24x7x365.

## EPM Workspace

Oracle Hyperion Financial Management is available within Oracle Hyperion Enterprise Performance Management Workspace. For information on tasks performed in EPM Workspace, such as preferences or features in the Navigate, Favorites, Manage, or Tools menu, see the *Oracle Hyperion Enterprise Performance Management Workspace User's Guide* and online help.

# Financial Management Dimensions

Dimensions describe an organization's data and usually contain groups of related members. Examples of dimensions are Account, Entity, and Period. Oracle Hyperion Financial Management provides eight system-defined dimensions and enables you to populate an unlimited number of custom dimensions that you can apply to accounts.

The elements that comprise a dimension are called members. For example, GrossMargin and TotalRevenues are members of the Account dimension.

Dimension members are arranged in hierarchies. Upper-level members are called *parent members*, and a member immediately below a parent member is referred to as its *child*. All members below a parent are referred to as *descendants*. The bottom-level hierarchy members are called *base-level members*.

Not all data dimensions allow you to define hierarchies. For example, the Year dimension does not support hierarchies.

Data is entered into base-level members of dimensions and not into parent members. Values for parent-level members are aggregated from the children of the parent-level members. In some cases, data for base-level members is calculated.

The following sections describe the system-defined dimensions.

## Scenario Dimension

The Scenario dimension represents a set of data, such as Actual, Budget, or Forecast. For example, the Actual scenario can contain data from a general ledger, reflecting past and current business operations. The Budget scenario can contain data that reflects the targeted business operations. The Forecast scenario typically contains data that corresponds to predictions for upcoming periods. A Legal scenario can contain data calculated according to legal GAAP format and rules.

You can define any number of scenarios for an application and define attributes for Scenario dimension members, such as the default frequency, the default view, and zero data settings.

## Year Dimension

The Year dimension represents the fiscal or calendar year for data. An application can contain data for more than one year. You specify a year range when you create the application and select a year from the Year dimension to process data.

## Period Dimension

The Period dimension represents time periods, such as quarters and months. It contains time periods and frequencies by displaying the time periods in a hierarchy. For example, if the Actual scenario maintains data on a monthly basis, generally 12 periods of data are available for this scenario in a year. Oracle Hyperion Financial Management supports years, months, and weeks for the Period dimension.

## Entity Dimension

The Entity dimension represents the organizational structure of the company, such as the management and legal reporting structures. Entities can represent divisions, subsidiaries,

plants, regions, countries, legal entities, business units, departments, or any organizational unit. You can define any number of entities.

The Entity dimension is the consolidation dimension of the system. Hierarchies in the Entity dimension reflect various consolidated views of the data. Various hierarchies can correspond to geographic consolidation, legal consolidation, or consolidation by activity. All relationships among individual member components that exist in an organization are stored and maintained in this dimension. Entities in an organization can be categorized as base, dependent, or parent entities. Base entities are at the bottom of the organization structure and do not own other entities. Dependent entities are owned by other entities in the organization. Parent entities contain one or more dependents that report directly to them.

You define attributes for Entity dimension members, such as the default currency and security class, and to specify whether the entity allows adjustments and stores intercompany detail.

## Value Dimension

The Value dimension represents the types of values stored in your application, and can include the input currency, parent currency, adjustments, and consolidation detail such as proportion, elimination, and contribution detail. For example, the Entity Currency member stores the value for an entity in its local currency. The Parent Currency member stores the value for an entity translated to the currency of the parent entity. The Value dimension is useful for providing an audit trail of the transactions applied to data.

## Account Dimension

The Account dimension represents a hierarchy of natural accounts. Accounts store financial data for entities and scenarios in an application. Each account has a type, such as Revenue or Expense, that defines its accounting behavior.

You define attributes for Account dimension members, such as the account type, the number of decimal places to display, and whether the account is a calculated, consolidated, or intercompany partner account.

## Intercompany Dimension

The Intercompany dimension represents all intercompany balances that exist for an account. This is a reserved dimension that is used in combination with the Account dimension and any custom dimension. Oracle Hyperion Financial Management can track and eliminate intercompany transaction details across accounts and entities. You can also run Intercompany Matching reports to view intercompany transactions.

## View Dimension

The View dimension represents various modes of calendar intelligence such as Periodic, Year-to-Date, and Quarter-to-Date frequencies. If you set the view to Periodic, the values for each month are displayed. If you set the view to Year-to-Date or Quarter-to-Date, the cumulative values for the year or quarter are displayed.

## Custom Dimensions

Custom dimensions are dimensions associated with accounts. These dimensions enable you to specify additional details associated with accounts, such as products, markets, channels, balance sheet movement, or types of elimination. For example, Custom dimensions could include Product Line, Region, Channel, or Customers. A Custom dimension for products

associated with Sales and COGS accounts enables you to track sales and cost detail by product.

## User-Defined Elements

Many elements in Oracle Hyperion Financial Management are user-defined. For example, when you create a journal, you specify a label and description.

User-defined elements, the minimum and maximum length for each element, and additional restrictions are listed below. The table groups the elements by the modules in which they are found.

**Table 1-1 Requirements for User-Defined Elements**

Element	Min. length	Max. length	Restrictions
Application Profile			
Language	1	20	None
Period label	1	80	<ul style="list-style-type: none"> <li>• Must contain only alphanumeric characters.</li> <li>• Cannot contain spaces, symbols, or diacritical marks such as umlauts.</li> </ul>
View label	1	10	<ul style="list-style-type: none"> <li>• Must contain only alphanumeric characters.</li> <li>• Cannot contain spaces, symbols, or diacritical marks such as umlauts.</li> </ul>
View description	0	40	Cannot contain an ampersand (&).
Period description	0	40	Cannot contain an ampersand (&).
Create Application			
Application label	1	10	<ul style="list-style-type: none"> <li>• Must contain only alphanumeric characters.</li> <li>• Cannot start with a number.</li> <li>• Cannot contain spaces, symbols, diacritical marks such as umlauts, or special characters such as German Capital Sharp S.</li> </ul> <p><b>Note:</b> Application labels are not case-sensitive. For example, App1 and APP1 are considered the same application label.</p>
Application description	1	255	<ul style="list-style-type: none"> <li>• Cannot contain an ampersand (&amp;).</li> <li>• Cannot contain spaces.</li> </ul>

**Table 1-1 (Cont.) Requirements for User-Defined Elements**

Element	Min. length	Max. length	Restrictions
Member label	1	80	<p>Must be unique. The label can contain up to 80 characters including spaces, but cannot start with a space.</p> <p>Cannot include these characters:</p> <ul style="list-style-type: none"> <li>• Ampersand (&amp;)</li> <li>• Asterisk (*)</li> <li>• At sign (@)</li> <li>• Comma (,)</li> <li>• Curly brackets ({} )</li> <li>• Double quotation mark ""</li> <li>• Forward slash (/)</li> <li>• Minus sign (-)</li> <li>• Number sign (#)</li> <li>• Period (.)</li> <li>• Plus sign (+)</li> <li>• Semicolon (;)</li> </ul> <p><b>Note:</b> You cannot use ALL as the name of an entity.</p>
Member description	0	40	<p>Cannot contain an ampersand (&amp;).</p> <p><b>Note:</b> Avoid using a period (.) in a member description when you select to work in Description mode.</p>
Alias label	0	80	<p>Cannot contain an ampersand (&amp;).</p>
Security class	1	80	<p>Cannot include these characters:</p> <ul style="list-style-type: none"> <li>• Asterisk (*)</li> <li>• At sign (@)</li> <li>• Comma (,)</li> <li>• Curly brackets ({} )</li> <li>• Double quotation mark ""</li> <li>• Minus sign (-)</li> <li>• Number sign (#)</li> <li>• Period (.)</li> <li>• Plus sign (+)</li> <li>• Semicolon (;)</li> <li>• Slash mark (/)</li> </ul>
Journals			



Table 1-1 (Cont.) Requirements for User-Defined Elements

Element	Min. length	Max. length	Restrictions
Journal label	1	20	Cannot include these characters: <ul style="list-style-type: none"> <li>• Asterisk (*)</li> <li>• At sign (@)</li> <li>• Comma (,)</li> <li>• Curly brackets ({})</li> <li>• Double quotation mark ""</li> <li>• Forward slash (/)</li> <li>• Minus sign (-)</li> <li>• Number sign (#)</li> <li>• Period (.)</li> <li>• Plus sign (+)</li> <li>• Semicolon (;)</li> </ul>
Journal description	0	255	None
Journal group	0	30	None
Journal line item description	0	50	None
Load/Extract			
Delimiter character	1	1	Must be one of these characters and cannot be used in the file or in the file name: <ul style="list-style-type: none"> <li>• Ampersand (&amp;)</li> <li>• At sign (@)</li> <li>• Backslash (\)</li> <li>• Carat (^)</li> <li>• Colon (:)</li> <li>• Comma (,)</li> <li>• Dollar sign (\$)</li> <li>• Line ( )</li> <li>• Percent sign (%)</li> <li>• Question mark (?)</li> <li>• Semicolon (;)</li> <li>• Tilde (~)</li> </ul> <p><b>Note:</b> You must use the same delimiter character throughout the file. Using different delimiter characters within the same file causes an error when you load the file.</p>
Data grids			
Cell description	1	1900	None
Line item detail	1	80	None
Annotation	0	255	None

Table 1-1 (Cont.) Requirements for User-Defined Elements

Element	Min. length	Max. length	Restrictions
Decimal character	1	1	These characters are invalid decimal characters for data grids: <ul style="list-style-type: none"><li>• Backslash (\)</li><li>• Forward slash (/)</li><li>• Minus sign (-)</li><li>• Plus sign (+)</li></ul>
Documents			
Document names (including folder and report names)	1	16	These characters are invalid characters for document names: <ul style="list-style-type: none"><li>• Asterisk (*)</li><li>• At sign (@)</li><li>• Backslash (\)</li><li>• Colon (:)</li><li>• Comma (,)</li><li>• Curly brackets ({})</li><li>• Double quotation mark ""</li><li>• Forward slash (/)</li><li>• Greater than sign (&gt;)</li><li>• Less than sign (&lt;)</li><li>• Line ( )</li><li>• Number sign (#)</li><li>• Period (.) at the end of a document name</li><li>• Plus sign (+)</li><li>• Question mark (?)</li><li>• Semicolon (;)</li></ul> <b>Note:</b> Document names also cannot contain trailing or leading white space.

# 2

## Managing Applications

An application consists of a set of entities, accounts, scenarios, and other dimensions that you use together. You can create as many applications as you need. For example, you can set up one application to report on tax data for several organizations and another application to report on Security and Exchange Commission data for other organizations.

For application administration, these security roles are required: Dimension Editor and Financial Management Application Creator/Financial Management Manager. For information on roles, see the *Oracle Enterprise Performance Management System User Security Administration Guide*.

Applications run on application servers. You can set up clusters of application servers to balance the load on multiple servers. For instructions, see the *Oracle Enterprise Performance Management System Installation and Configuration Guide*.

Only members of the Administrator group that you specify when you configure the application server can perform these administrative tasks:

### Users on System

- List users
- Log out users

### Manage Servers and Applications

- View disabled components
- Enable or disable connections
- Log out users

### System Messages

- View - must be member of Administrator group
- Delete - must be member of Administrator group

To assign users to the Administrator Group, see the *Oracle Enterprise Performance Management System User Security Administration Guide*.

## Application Administration

You define an application using these steps:

1. Create an application shell, in which you specify the application server where the application resides, an application label and description, calendar, language, and frequency properties. See [Creating Applications](#).
2. Define security for the application, including which users have access to the application and what type of access each user has. See [Managing Application Security](#).
3. Define metadata for the application, including accounts, entities, scenarios, and custom dimensions, application settings, consolidation methods, and currencies. See [Managing Metadata](#).

4. Load data, data forms, member lists, rules, and journals to the application.

## Creating Applications

An application is a set of dimensions and dimension members that meet a set of analytical or reporting requirements. For example, you can have an application to report on tax data and a separate application for sales analysis.

For detailed information on Oracle Hyperion Tax Provision applications, see the Tax Provision documentation.

There are several methods for creating applications:

- Create a new application. This option launches the Create Application wizard. You can define the application calendar, module configuration, and custom dimensions. See [Creating a New Application](#).
- Copy from an application. View the application calendar, module configuration and custom dimensions from an application and modify using the Application Creation wizard to create a new application. See [Copying from an Application](#).
- Create an application from a profile file. You can also view and modify the calendar. See [Creating an Application from a Profile File](#).

## Creating a New Application

To create a new application, you must have the Application Administrator security role.

To create an application:

1. Select **Navigate**, then **Administer**, and then **Consolidation Administration**.
2. From the **Applications** tab, select **Actions**, and then **New**.

The system launches the Application Creation wizard.

3. From **Application options**, select **New application**, and then click **Next**.
4. In the **Application Properties** section, for **Name**, enter a name for the new application.

The application label can have a maximum of either 10 alphanumeric characters or 12 bytes. It cannot start with a number or contain spaces, an ampersand (&) symbol, special characters, or more than five international characters. It also cannot contain these characters: German capital ſ (Unicode U+1E9E) and lowercase ß (Unicode U+00DF), or Turkish dotless lower case: ı (Unicode U+0131).

Application labels are not case-sensitive. For example, App1 and APP1 are considered the same application. HFM, HSV, and HSX are reserved names and cannot be used for application labels or descriptions.

### **Caution:**

Do not create applications with the same name even if they are on different Oracle Hyperion Financial Management application servers. Applications of the same name but from different Financial Management application servers cannot coexist on one Oracle Hyperion Shared Services server.

5. For **Description**, enter a description for the application.

The application description can have a maximum of 255 characters, and can include spaces. It cannot contain an ampersand (&). HFM, HSV, and HSX are reserved names and cannot be used for application descriptions.

6. From the **Application Type** list, select a type:
  - **Consolidation**
  - **Tax Provisioning**
7. From the **Cluster** list, select the application server cluster on which to run the new application.

If the server cluster is not listed, you may need to register it. See the *Oracle Enterprise Performance Management System Installation and Configuration Guide*.
8. From the **User Management Project** list, select the Shared Services project to which to add the application.

 **Note:**

Each application must belong to a project. See the *Oracle Enterprise Performance Management System User Security Administration Guide*.

9. For **Languages**, you can specify up to 10 languages for descriptions that are used throughout an application. Use a comma to separate the language names. Each language name can contain a maximum of 20 characters. Note that a space is counted as a character.

 **Caution:**

After you create an application, you cannot change the languages in the application.

10. Define a calendar using the Year, Period, and View dimensions. The members in the Year dimension define the start year and the total number of years in the application. The member hierarchy in the Period dimension defines the base-level and summary-level time periods. The View dimension defines the period-to-date views, such as year (YTD), half-year (HYTD), quarter (QTD), and month (MTD).

 **Caution:**

After you create an application, you cannot change the start year, period descriptions or labels in the application.

- In the **Years** section, for **Start Year**, enter the start year for the calendar.
- For **Number of Years**, enter the total number of years to include in the application.

 **Note:**

The number of years in an application is the only profile definition that can be modified for existing applications. See [Modifying Applications](#).

11. In the **Time Periods** section, select the time periods to include:

- **Months**

Select the time periods to include (**Half-Years**, **Quarters**, or **Trimesters**)

From the **Start Month** drop-down list, select the first month in the calendar.

- **Quarters**

**Optional:** Select to include **Half-Years**.

- **Custom.** If you choose to define a Custom calendar, you must define the period label prefix and number of base periods. A flat list is created and you can modify the hierarchy later.

- For **Period Label Prefix**, enter a prefix for the periods to include.

The label can contain a maximum of 10 characters and can include spaces.

- For **Number of Base Periods**, enter the number of periods in the year.

If you enter any number less than 10 for the number of periods and NewPeriod as the label prefix, these periods are added to the hierarchy: NewPeriod1, and so on. If the number of periods is more than 10 but less than 100, the periods start with NewPeriod01. If the number of periods is more than 100 but less than 1000, the periods start with 001, and so on.

 **Note:**

If you choose to define a Manual calendar, leave the period label prefix and number of base periods blank. You can then enter the frequencies and periods that you need.

12. Click **Next**.

## Defining Frequencies

The frequency specifies the time period level at which you can input data, for example, months or quarters. Frequencies and their corresponding views (YTD, HYTD, QTD, MTD) are created based on the time periods that you selected when defining the calendar. You can add, modify, and delete frequencies. You can also enter a descriptive label for each frequency and view in each language that you previously defined.

Year-to-Date (YTD) is provided by default in the first row of the Frequency column. You cannot delete or change the YTD label. However, you can enter a YTD description for each language that you define.

 **Note:**

In addition to the frequencies that you can define, each application contains two system-defined frequencies and corresponding views: Scenario View and Periodic.

If you selected to define a calendar manually, the Frequencies grid is empty, and you must enter the necessary views and their descriptions. You should enter one frequency for each level of the Period dimension.

 **Caution:**

After you create an application, you cannot change the frequency descriptions or labels in the application.

To define frequencies:

1. From the **Frequencies** screen, click the **(+) Add Frequency** button, or select **Actions**, and then **Add Frequency**.
2. Enter one or more frequencies and descriptions.

By default, YTD is the first frequency. The number of frequencies that you define should be equal to the number of generations in the Period tree.

 **Note:**

The label can contain a maximum of 40 characters. The description can contain a maximum of 80 characters.

Labels cannot include these characters: + - \* / # { } ; , @ ""

 **Tip:**

Because you cannot modify frequencies after an application is created, make sure to include a description for each frequency in each language.

3. Click **Next**.

## Editing Periods

The period hierarchy is a combination of the time period and frequencies that you define. You can make changes to this hierarchy by adding or deleting periods. For example, you may want to add another month to the fourth quarter to display a 13-month year.

 **Note:**

The number of base periods should be greater than 0.

When you select a period, you can change the period label or description. You cannot edit the label or description for the [Year] period.

 **Note:**

You cannot edit period labels based on language. If you change a period label in one language, the change is carried over into all other languages defined in the application profile. However, you can have a unique period description for each language.

To add sibling and child periods to the hierarchy, see [Adding Sibling and Child Periods](#).



To enter and edit periods:

1. From the period hierarchy, select a period, and enter or edit the period label or description.  
The period label can contain a maximum of 40 characters. The period description can contain a maximum of 80 characters. Note that a space is counted as a character.
2. Make sure to include a description for each period.  
If you do not include a description and choose to add one later, you must modify the application profile and recreate all applications that use the profile.


## Adding Sibling and Child Periods

You can add one or multiple sibling and child periods to the period hierarchy.

To add one child or sibling period:

1. Highlight the period to which to add a child or sibling period.
2. Select an option:
  - Click the **Insert Child** button, , or select **Actions**, and then **Insert Child**.
  - Click the **Insert Sibling** button, , or select **Actions**, and then **Insert Sibling**.
3. Enter a name for the new sibling or child period.

To add multiple child or sibling periods:

1. Highlight the period to which to add multiple child or sibling periods.
2. Click the **Insert Many** button, , or select **Actions**, and then **Insert Many**.
3. Enter the number of periods to add to the hierarchy.
4. Enter a label prefix for the new periods, and click **OK**.

If you enter any number less than 10 for the number of periods and NewPeriod as the label prefix, these periods are added to the hierarchy: NewPeriod1, and so on. If the number of



periods is more than 10 but less than 100, the periods start with NewPeriod01. If the number of periods is more than 100 but less than 1000, the periods start with 001, and so on.

5. When you finish adding periods, click **Next**.

## Deleting Periods

When you delete periods from the hierarchy, all descendants of the period are also deleted.



### Note:

You cannot delete the [Year] period.

To delete a period:

1. Select a period to delete.
2. Click **Delete Period** or select **Actions**, and then **Delete Period**.

Click **Next**.

## Defining Features

The Feature screen lists the Application Settings and default values based on the application modules that are enabled. You can change any of the default values.

## Specifying Application Settings

For detailed information on Application settings, see [Defining Application Settings](#).

You can specify these settings:

- **Application Currency** - The currency to use in the application. You can specify a currency and it will be added to the Currency dimension. See [Defining Currencies](#).
- **Rate Account for Balanced Accounts** - The rate account for BALANCE accounts. You can specify an account and will be added to the Account dimension. See [Defining Accounts](#).
- **Rate Account for Flow Accounts** - The rate account for FLOW accounts. You can specify an account and will be added to the Account dimension. See [Defining Accounts](#).
- **Translate Method for Balanced Accounts** - The translation method for BALANCE accounts. See [Defining Accounts](#).
- **Translate Method for Flow Accounts** - The translation method for FLOW accounts. See [Defining Accounts](#).

To specify application settings:

1. From the **Features** screen, specify values for these Application Settings, or use the default values:
  - **Application Currency** - Enter a currency for the application. All currencies are listed and available to select. For example, if you type US, the USD - US Dollar currency is displayed and available to select.

- **Rate Account for Balanced Accounts** - Enter the rate account to use for BALANCE accounts.  
The account name can contain up to 80 characters, including spaces, but cannot start with a space. See [Defining Accounts](#).
  - **Rate Account for Flow Accounts** - Enter the rate account to use for FLOW accounts.  
The account name can contain up to 80 characters, including spaces, but cannot start with a space. See [Defining Accounts](#).
  - **Translate Method for Balanced Accounts** - From the dropdown list, select the translation method for BALANCE accounts:
    - PVA - for the periodic value translation method
    - VAL - to use the value at exchange rate translation method
  - **Translate Method for Flow Accounts** - From the dropdown list, select the translation method for FLOW accounts:
    - PVA - for the periodic value translation method
    - VAL - to use the value at exchange rate translation method
2. To enable or disable application modules, see [Enabling or Disabling Application Modules](#).

## Enabling or Disabling Application Modules

When you create an application, all application modules are enabled by default. Users can view modules for which they have the appropriate security role.

Depending on the application type, you might not require all modules. If you are the Application Administrator, you can select to disable specific application modules. For example, you could disable the Journals or Intercompany Transactions modules for certain applications. When you disable a module, it is not displayed for any application user.



### Note:

If you have documents associated with a module that you want to disable, such as journal or intercompany transaction reports, make sure to also remove them from the system.

After you make changes to modules, close and reopen the application to see the changes.

The application module configuration information is saved as an XML file named `application_name_Moduleconfig.xml`. You can load and extract module configuration information from the Load Application Elements and Extract Application Elements pages.

To enable or disable application modules:

1. From the **Features** screen, Select **Yes** to enable or **No** to disable application modules, or use the default values:
  - **Enable Process Control**

 **Note:**

If you do not enable Process Control, you cannot enable Submission Phases.

- **Enable Submission Phase**
- **Enable Manage Ownership**
- **Enable Journals**
- **Enable Data Management**
- **Enable Audit Tasks**
- **Enable Intercompany Transactions**
- **Enable Equity Pickup**

2. Click **Next**.

## Creating Custom Dimensions

You use the Dimensions screen to configure dimensions and security properties for dimensions. You can add or delete Custom dimensions, and specify a Custom Dimension name and Alias.

In the Dimensions table, the first two Custom dimensions for the application are automatically displayed. These Custom dimensions are used for currency rate and consolidation method information, and the size for these dimensions must be Large. The first dimension is used for "From Currency" and consolidation method information, and the second dimension is used for "To Currency" information. You can specify the dimension name (short label) and dimension alias (long label) for the Custom dimension for these two entries. You cannot change the size, and you cannot delete these dimensions.

In addition to the default dimensions, you can create additional Custom dimensions with these guidelines:

- You must enter a unique Custom dimension name and dimension alias for the application. Both the name and alias must be unique. For example, if you have a dimension name of PROD, the dimension alias cannot be PROD. The Custom dimension name also cannot be the same as a Currency name.
- The dimension name can be a maximum of 10 characters.
- The dimension alias can be a maximum of 20 characters.
- The dimension name and alias cannot contain spaces, and cannot be blank.
- You must select a size for the Custom dimension: Small, Medium, or Large.

The maximum number of Custom dimensions depends on the database type and the size of each Custom dimension.

- The Large Custom dimension supports 2 billion members, and requires 4 bytes of space in a database column for storage.
- Medium supports 32,000 members and requires 2 bytes.
- Small supports up to 128 and requires 1 byte.

The calculation method of the maximum number of Custom dimensions is the same regardless of the database type, but the database type determines the amount of total space available.

The total number of physical Custom columns dictates the total number of bytes available for Custom dimension storage in Oracle, Microsoft SQL Server, or IBM DB2 databases.

- Oracle supports up to 21 physical Custom columns; 21 x 8 = 168 bytes total available space.
- SQL Server supports 5 physical Custom columns; 5 x 8 bytes = 40 bytes total available space
- IBM DB2 supports primary keys up to 900 bytes, = 800 bytes total available space

At least two Custom dimensions are required. Below are the formulae for calculating more than two Custom dimensions:

Migrating applications:  $4 \times LD + 2 \times MD + SD \leq \text{MAXBYTES} - 8$

New applications:  $4 \times LD + 2 \times MD + SD \leq \text{MAXBYTES}$

*LD* = number of large dimensions, *MD* = number of medium dimensions, and *SD* = number of small dimensions.

MAXBYTES = 40 for Microsoft SQL Server, 168 for Oracle, 800 for IBM DB2

Oracle recommends that you use Medium for Custom dimensions for all practical purposes.

The Small size for Custom dimensions is not recommended if you anticipate having more than 128 members over the lifetime of the application.

Oracle recommends that you use the Large size for Custom dimensions only if you plan to have more than 32,536 members in that dimension over the lifetime of the application.

The following example shows sample Custom dimensions.

Custom Dimension Name	Custom Dimension Alias	Custom Dimension Size	Use for Currency
Prod	Products	Large	From Currency
Mkt	Markets	Large	To Currency
Flows	BSFlows	Small	N/A
Cust	Customers	Large	N/A

To create Custom dimensions:

1. From the **Dimensions** table, click the **(+) Create Custom Dimension** button, or select **Actions**, and then **Create Custom Dimension**.
2. For **Short Name**, enter a unique dimension name.
3. For **Alias Name**, enter a unique dimension alias.
4. For **Dimension Size**, select an option: **Small**, **Medium**, or **Large**.
5. To create additional dimensions, repeat these steps.
6. **Optional:** If you have completed all the steps to create an application, click **Create Application**.

When the application is successfully created, a confirmation message displays that the application has been created and that the metadata has been loaded successfully. From the confirmation message, click **OK**.



**Note:**

After you create an application, you cannot modify the number of dimensions. If you need additional Custom Dimensions, you must create a new application with additional Custom Dimensions.

## Saving Application Profiles

After you complete all the steps in the Application Creation Wizard and click Create Application, the system generates a binary application Profile (.PER) file using the options that you selected during the wizard navigation. The application is then created using the Profile file. Metadata is updated and loaded with your selected settings.

After a Profile file is created for an application, you can create a new application using the Profile file. See [Creating an Application from a Profile File](#).

## Creating an Application from a Profile File

When you create a new application, the system generates a binary application Profile (.PER) file using the options that you selected during the wizard navigation. The application is then created using the Profile file. You can create a new application using the Profile file from an existing application.

When you use a Profile file, the system uses the default values for application settings.

To create an application from a Profile file:

1. Select **Navigate**, then **Administer**, and then **Consolidation Administration**.
2. From the **Applications** tab, select **Actions**, and then **New**.  
The system launches the Application Creation wizard.
3. From **Application options**, select **Application from Profile File**, and then click **Next**.
4. From the **Browse Profile File** screen, click **Browse**, and select an application Profile (.PER) file, and then click **Next**.
5. Modify the Application Properties if needed, and then click **Create Application**.

## Copying from an Application

To create an application, you can copy an application profile from an existing application. You can view the application calendar, module configuration and custom dimensions from an application and modify using the Application Creation wizard to create a new application.

To create an application:

1. Select **Navigate**, then **Administer**, and then **Consolidation Administration**.
2. From the **Applications** tab, select **Actions**, and then **New**.  
The system launches the Application Creation wizard.
3. From **Application options**, select **Copy from an application**, and then click **Next**.
4. From the **Select Application** screen, select an application to copy.

The wizard is populated with the information from the selected application, including application settings and module configuration. You can change the current values.

- To create an application from the Application Properties screen, click **Create Application**.
- To modify application information, click **Next** to navigate through the wizard.

## Modifying Applications

After you create an application, modifications to the Application Profile are generally not allowed. However, an administrator can change these settings:

- Enable or disable application modules. See [Enabling or Disabling Application Modules](#).
- Increase the number of years used in the application.

You can modify the number of years the application supports, with these limitations:

- The new profile end year should exceed the current end year. You can only increase the number of years; decreasing the number is not supported.
- The new end year must be less than 2100.
- You cannot modify the application profile Start Year.

### Example

Current start year: 2005

Current number of years: 10 (End year 2015)

Request for modification: 15 (End year 2020)

Before the application profile end year can be processed, the system sets the application into Admin Mode. The current application process is stopped and all users logged into the application are forcefully logged out.

To modify an application:

1. Select **Navigate**, then **Administer**, and then **Consolidation Administration**.
2. From the **Applications** tab, select an application to modify.
3. Click **Modify Application**, or select **Actions**, and then **Modify Application**, or right-click on an application and select **Modify Application**.

Note that most of the Application Properties are grayed out.

4. To change the **Number of Years**, enter the number of years for the application.
5. Click **Modify Application**.

The system displays a warning message before processing the modified application. If you click Yes, it sets the application to Admin Mode, and validates the number of years. If there are no validation errors, it updates the Year dimension, and then disables the Admin Mode.

## Registering Applications

During the installation process, you configure and register server clusters. After you create an application, you can register the application against the preferred server cluster.

To register applications:

1. Select **Navigate**, then **Administer**, and then **Consolidation Administration**.
2. Select **Administration**, and then **Applications**.
3. From the application list, select an application.
4. Click **Register**, or select **Actions**, and then **Register**.
5. From the **Cluster** list, select a server cluster.
6. From the **User Management Project** list, select the Oracle Hyperion Shared Services project.
7. Click **Register**.

## Opening Applications

In Oracle Hyperion Financial Management, all data is processed within applications. You can open and work with multiple applications at one time.

To access an application, you must be assigned as a user of the application.

To open an application:

1. Select **Navigate**, then **Applications**, then **Consolidation**.
2. Select an application.

 **Note:**

If the application is not listed, click **Refresh**.

## Closing Applications

You can close the current application in which you are working, or if you have multiple applications open, close them all simultaneously.

To close an application, take one of these actions:

- Select **File**, then **Close**, and then **Current**, or **All**.
- Click the X on the top of the tab in which the application is open.

## Changing Application Server Clusters

You use the Administration module to manage applications. By default, the Administration module uses the first available cluster/server. You can change the connected server using this option, which reloads the Administration module.

To change the server cluster:

1. Select **Navigate**, then **Administer**, then **Consolidation Administration**.
2. Select **Administration**, and then **Change Cluster**.
3. From the Cluster Server list, select a cluster.
4. Click **OK**.

## Viewing the Application List

The Applications table contains a list of all of the available Oracle Hyperion Financial Management applications in the system. The table displays application names and descriptions.

To view the application list:

1. Select **Navigate**, then **Administer**, and then **Consolidation Administration**.
2. Select **Administration**, and then **Applications**.

## Deleting Applications

Before you delete an application, ensure that no other user is currently using the application.

To delete an application, you must have these security roles:

- Shared Services: Financial Management Manager, or Shared Services: Financial Management Administrator and Shared Services: Financial Management Application Creator
- Application: Application Administrator
- Application: Provisioning Manager

To delete an application:

1. Select **Navigate**, then **Administer**, and then **Consolidation Administration**.
2. Select **Administration**, and then **Applications**.
3. Select the application to delete.
4. Click **Delete**, or select **Actions**, and then **Delete**.
5. If the application is open, a warning message displays. Select an option:
  - To delete the application forcefully, click **Force Delete**.  
Users who are logged in receive error messages after the application is deleted.
  - To view the application users, click **See Users**. From the System Users page, you can log out users.
  - To cancel application deletion, click **Cancel**.
6. If the application is not open by you or other users, a confirmation prompt displays. Click **Yes** to delete the application.

## Loading Application Elements

After you create an application, you load metadata, member lists, rules, and security files. You can load individual files, select multiple files to load, or load all files at once. When you load



multiple files, the system loads them in the proper sequence. You can also scan the files to verify them before loading.

Each load process generates a log file so you can review each process individually. When the load process is complete, a link displays for the log so that you can view any errors. If one of the load files does not complete successfully, you can correct any errors and reload it.

Application element files must use a specific file format, and several options are available for the load process. See these sections:

- [Loading Application Security](#)
- [Loading Metadata](#)
- [Loading Member Lists](#)
- [Loading Rules](#)

To load application elements:

1. Open an application.
2. Select **Consolidation**, then **Load**, and then **Application Elements**.
3. Enter the file name to load, or click **Browse** to locate the file that you want to load.
4. Specify options for the load process.
5. **Optional:** Click **Scan** to verify that the file format is correct.
6. Click **Load** to load individual files or click **Load All**.

If you reload existing files, the system displays a warning prompt asking if you want to use them again. If you do, click **Yes**.

 **Tip:**

To reset the file options to the default values, click **Reset** or **Reset All**.

7. **Optional:** To download the log file, click **Download Log**. Click **Open** to display the log file, or **Save** and select a location to save the file locally.

## Extracting Application Elements

You can extract application elements, view and modify the information in a text editor, and then reload the elements into the application. This can be useful if you need to make updates to multiple files simultaneously. You can extract individual files, select multiple files to extract, or extract all files at once.

Application element files must use a specific file format, and several options are available for the extract process. See these sections:

- [Extracting Application Security](#)
- [Extracting Metadata](#)
- [Extracting Member Lists](#)
- [Extracting Rules](#)

When the extract process is complete, a link displays for the log so that you can view any errors.

To extract application elements:

1. Open an application.
2. Select **Consolidation**, then **Extract**, and then **Application Elements**.
3. Specify options for the extract process.
4. Click **Extract** to extract individual files or click **Extract All**.
5. Click **Download** to download the extracted file.
6. **Optional:** To download the log file, click **Download Log**. Click **Open** to display the log file, or **Save** and select a location to save the file locally.

## Duplicating Financial Management Applications

You can duplicate the contents of an application to create a new application. To duplicate an application, you must have the Application Administrator role, and be provisioned in Shared Services for the Financial Management Application Creator role.

You can duplicate an application to the same environment and rename it, or you can register the new application in a different project and cluster. You can only duplicate an application in the same database.

By default, the process duplicates the application profile information, metadata, and security information. You can also select to duplicate data tables and the Task Audit and Data Audit tables from the source application to the destination application.

To duplicate an application:

1. Select **Navigate**, then **Administer**, and then **Consolidation Administration**.
2. Select **Administration**, and then **Applications**.
3. Select the application that you want to duplicate.
4. Click **Duplicate Application**, or select **Actions**, and then **Duplicate Application**, or right-click the application name and select **Duplicate Application**.
5. For **Name**, enter the application name of the new application that you want to create.
6. For **Description**, enter an application description.
7. For **Cluster**, select the cluster on which the application should be registered.
8. For **User Management Project**, select the project on which the application should be registered.
9. **Optional:** Select the **Duplicate Audit Tables** option to duplicate the Data Audit and Task Audit tables from the source to the destination application.
10. **Optional:** Select the **Duplicate Data Tables** option to duplicate the data tables from the source to the destination application.
11. Click **Duplicate**.

When you start the duplication process, the Admin mode is enabled. All users are logged out of the application and are not allowed to log in again until the process is completed.

The system displays the Admin Tasks page, where you can monitor the progress of the Duplicate Application task.

After the system successfully duplicates the source application, it registers the newly created application to the cluster and project that you specified, and you can open the new application.

# Importing Applications

You can import an application across databases or across EPM systems using homogeneous databases, however both EPM systems must use either Oracle Database or SQL Server. Source applications can originate from prior releases as far back as 11.1.2.4, however to upgrade older applications, you must run the Financial Management Upgrade Application task in **EPM System Configurator** from one Financial Management server. To import an application, you must have the Application Administrator role and be provisioned in Shared Services for the Financial Management Application Creator role.

You can import an application to the same environment and rename it, or you can register the new application in a different project and cluster.

You can import data for all years and scenarios, or select specific years and scenarios.

By default, the process duplicates the application profile information, metadata, and security information. You can also select to import the task and data audit information from the source application to the destination application.

## Prerequisites for Importing Applications

These steps are required before you import applications:

- You must create a data communication link between the source and destination databases.
- If you are using Oracle databases, you must create a Database Link.
- If you are using SQL Server databases, you must create a Linked Server.
- You must make sure that the HFM database has the rights to create a Database Link or Linked Server.
- You must grant a one-time explicit DB privilege for database users configured in the Database Link or Linked Server rather than provisioning the users through roles.

To import an application:

1. Log on to the destination EPM system.
2. Select **Navigate**, then **Administer**, and then **Consolidation Administration**.
3. Select **Administration**, and then **Import Applications**.
4. Select **Database Link** if you are importing between Oracle databases, or select **Linked Server** if you are importing between SQL Server databases.

When you select Database Link or Linked Server, the source applications are listed.

5. From the application list, select the source application.
6. Specify the **Destination Application Name** and **Destination Application Description**.  
The source application is imported with this name and description.
7. For **Cluster**, select the cluster the application should be registered on.
8. For **User Management Project**, select the project the application should be registered on.
9. **Optional:** Select the **Import Audit Tables** option to import the Data Audit and Task Audit tables from the source to the destination application.
10. **Optional:** Select the **Import Data Tables** option to import the data tables from the source to the destination application.

If you select **Import Data Tables**, you may also specify the Scenario and Year filters.

11. **Optional:** Select the **Purge Source Data** option to purge the source data tables after copying to the target application.
12. Click **Import**.

When you start the import process, the system displays a warning that the operation is data intensive and asks if you want to continue.

It then displays the Admin Tasks page, where you can monitor the progress of the Import Application task.

After the system successfully imports the source application, it registers the newly created application to the cluster and project that you specified, and you can open the new application.

## Creating a Database Link for an Oracle Database

Before you import an application, you must create a database link between the source and destination databases.

Perform these steps from the destination database.

To create a database link to source Oracle DB, HFM Schema:

1. Log in as sysdba user and Grant privileges to create a database link to HFM Schema.

```
GRANT CREATE DATABASE LINK TO hfm;
```

2. Log in to HFM Schema of the destination system and perform the following command:

```
CREATE DATABASE LINK <link name> CONNECT TO<hfm schema name>IDENTIFIED BY  
HFM1 USING '///host name:<port>/<service name>';
```

For example, to connect HFM Schema on host **SLCK58001**, and Oracle running on port **1521**:

```
CREATE DATABASE LINK ToTestSystem CONNECT TO HFM IDENTIFIED BY HFM1 USING  
'///slck58001.xxx:1521/service name';
```

3. Validation step: The following command should list the applications in the source system:

```
Select * from HSX_DATASOURCES@ToTestSystem
```

4. Command to drop database link:

```
drop database link ToTestSystem;
```

5. Command to list all database links:

```
select * from all_db_links
```

## Creating a Linked Server for a SQL Server Database

Before you import an application, if you are using a SQL Server database, you must create a linked server between the destination and source database.

Perform these steps from the destination database.

To create a linked server to source SQLServer instances:

1. Execute the following four commands in order (each starting with exec) on the destination HFM database:

```
exec sp_addlinkedserver N'<linked server name>',@srvproduct=N'',
@provider=N'SQLNCLI', @datasrc=N'<host name>'
```

```
exec sp_addlinkedsrvlogin N'<linked server
name>',@useself=N'FALSE',@rmtuser=N'<hfm db user>',@rmtpassword=N'<hfm db
password>'
```

```
exec sp_serveroption @server=N'<linked server
name>',@optname=N'rpc',@optvalue=N'true'
```

```
exec sp_serveroption @server=N'<linked server name>',@optname=N'rpc
out',@optvalue=N'true'
```

Ensure that the <hfm db user> has database owner access on the target database.

For example, to create a linked server with the name **ToTestSystem** to sqlserver on **stm10082**, and the source sqlserver HFM database credentials are **hfm/hyperion**:

```
exec sp_addlinkedserver
N'ToTestSystem',@srvproduct=N'',@provider=N'SQLNCLI',@datasrc=N'STM10082.xx
x.COM'
```

```
exec sp_addlinkedsrvlogin
N'ToTestSystem',@useself=N'FALSE',@rmtuser=N'hfm', @rmtpassword=N'password'
```

```
exec
sp_serveroption@server=N'ToTestSystem',@optname=N'rpc',@optvalue=N'true'
```

```
exec sp_serveroption @server=N'ToTestSystem',@optname=N'rpc
out',@optvalue=N'true'
```

2. Validation step: The following command should list the applications in the source SQL Server:

```
select * from [ToTestsystem].[hfm].[dbo].HSX_DATASOURCES;
```

3. Command to drop linked server:

```
sp_dropserver'<linked server name>', 'droplogins'
```

Example: sp\_dropserver '**ToTestSystem**', 'droplogins'

4. Command to list all linked servers:

```
sp_linkedservers
```

## Working with System Messages

The System Messages log displays a list of Oracle Hyperion Financial Management system messages. From the log, you can view the message summary, view details of individual messages, and print messages. The system displays an **X** for error messages, and **i** for information messages. You can delete system messages from the log. Messages remain in the log until you delete them.

To view system messages, you must be a member of the Administrator Group that you specify when you configure the application server. To assign users to the Administrator Group, see the *Oracle Enterprise Performance Management System User Security Administration Guide*.

See these procedures:

- [Viewing System Messages](#)
- [Deleting System Messages](#)

## Viewing System Messages

To view system messages:

1. Select **Navigate**, then **Administer**, and then **Consolidation Administration**.
2. Select **Administration**, then **System**, and then **Messages**.
3. **Optional:** From the **Filters** panel, select filter criteria:
  - **Application**
  - **Server**
  - **Date Range**
    - To view all dates, select **Include All**.
    - To specify a date range, select **Range**, and then specify **Minutes**, **Hours**, **Days**, or **Months**.
    - To specify specific dates, select **Custom**, and then enter a **Start** and **End** date, or click the calendar icon to select dates.
4. From the list of system messages, select a message for which to view detail.
5. Double-click the message to open it, or click **View**, or select **Actions**, and then **View**.
6. When you finish viewing system message details, click **OK**.

## Deleting System Messages

You can delete system messages from the System Messages log if you are assigned to the Administrator group.

To delete system messages:

1. Select **Navigate**, then **Administer**, and then **Consolidation Administration**.
2. Select **Administration**, then **System**, and then **Messages**.

3. From the list of system messages, select one or more messages to delete.
4. Take one of these actions:
  - To delete selected messages, click **Delete Selected** or select **Actions**, and then **Delete Selected**.
  - To delete messages based on the current filter settings, click **Delete based on the current filter settings**, or select **Actions**, and then **Delete Filtered**.
  - To delete all system messages, click **Delete All** or select **Actions**, and then **Delete All**.

**Tip:**

To refresh the list of messages, click **Refresh** or select **Actions**, and then **Refresh**.

## System Message Detail Strings

Some processes return strings of technical information for system messages such as errors. The strings contain a uniquely identifying error reference number, followed by various fields of information. The fields are delimited by semicolons, and each field has a label that is followed by a colon, as in this example:

```
Error Reference Number: {219EB33B-BF50-11D6-A43E-0000863DCCF1}
```

```
Num: 0x800415c6; Type: 1; DTime: 1/3/12 12:20:10 PM; Svr: SERVER1; File:
CHsxServerImpl.cpp; Line: 1842; Ver: 3.0.0.196;
```

The following table describes the system message fields:

**Table 2-1 Fields in System Message Detail Strings**

Field	Description
Num	Error number in hexadecimal form
Type	<i>For internal use only</i>
DTime	Error TimeStamp
Svr	Machine name of the computer on which the error occurred
File	Name of the source code file to which the error applies
Line	Line number in the source code file to which the error applies
Ver	Version number of the DLL to which the error applies

## Managing Application Access

You can use the Manage feature to control these access-related settings for an application:

- Log out all users for a specified application. See [Logging Out Users](#).
- Enable and disable application access for all users. See [Disabling and Enabling Connections](#).

Only members of the Administrator group that you specify when you configure the application server can manage access-related settings. To assign users to the Administrator Group, see the *Oracle Enterprise Performance Management System User Security Administration Guide*.

## Logging Out Users

To perform a system-wide process such as a backup and restore, you can log all users off an application or an application server.

Only members of the Administrator group that you specify when you configure the application server can log out users.

The logout action forcefully logs out all users on the selected application.



### Note:

To log out selected users, see [Managing System Users](#).

To log all users off an application or server:

1. Select **Navigate**, then **Administer**, and then **Consolidation Administration**.
2. Select **Administration**, then **System**, and then **Manage**.
3. Click **Logout All Users**, or select **Actions**, then **Logout All Users**, and then click **Yes** from the confirmation prompt.
4. From the Logout Result dialog box, click **OK**.

## Disabling and Enabling Connections

When you disable connections, the system prevents new users from logging on to the specified server or application. You can use the disable connections feature with the log out users feature. For example, you can disable logging on to an application, log out users logged on to the application, load metadata, and then enable connections to the application.

These options are available for user access:

- **Disable Connections** - This option disables connections for all users, including the Administrator. No users are allowed to access the selected application.
- **Enable Connections for Admin Only** - Application access is only allowed for the Administrator.
- **Enable Connections for All Users** - Application access is allowed for all users.

To disable or enable user connections to an application or server:

1. Select **Navigate**, then **Administer**, and then **Consolidation Administration**.
2. Select **Administration**, then **System**, and then **Manage**.
3. Take one of these actions:
  - Click **Disable Connections**, or select **Actions**, and then **Disable Connections**.



- Click **Enable Connections for Admin Only**, or select **Actions**, and then **Enable Connections for Admin Only**.
  - Click **Enable Connections for All Users**, or select **Actions**, and then **Enable Connections for All Users**.
4. To refresh the list of connections, click **Refresh**, or select **Actions**, and then **Refresh**.

## Viewing Application Connection Status

You can view connection status by a list of servers, or by a list of applications.

When you view by server, the system displays the server name and a list of the applications applicable for that server with their status: Enabled or Disabled.

When you view by applications, the system displays the application names, the server on which they reside, and their status: Enabled or Disabled.

To view application connection status:

1. Select **Navigate**, then **Administer**, and then **Consolidation Administration**.
2. Select **Administration**, then **System**, and then **Manage**.
3. Use the menu icon to toggle between display types:
  - If the list is displayed by Server, select **Click to view by applications**.
  - If the list is displayed by Applications, select **Click to view by servers**.

## Managing System Users

The System Users feature enables you to view the users on the system and log off users of an application or server. You can view which modules are being used by users and what activities are being performed. See [Viewing Users](#) and [Logging Out Users](#).



### Note:

To view users on system, you must be assigned the Financial Management Administrator security role.

Only members of the Administrator group that you specify when you configure the application server can log out users. To assign users to the Administrator Group, see the *Oracle Enterprise Performance Management System User Security Administration Guide*.

## Viewing Users

You can view logged-on users for all application and servers, or filter the list to view users of specific servers and applications.



### Note:

To view users on system, you must be assigned the Financial Management Administrator security role.

This information is available for each logged-on user:

- User name
- Current module in which the user is working
- Current activity of user
- Time the activity was started
- Server name
- Application name
- Status

To view logged-on users:

1. Select **Navigate**, then **Administer**, and then **Consolidation Administration**.
2. Select **Administration**, then **System**, and then **Users**.
3. **Optional:** From the **Filters** panel, filter users by Application, or by Server.

## Logging Out Selected or All Users

To perform a system-wide process such as a backup and restore, you can log users off an application or an application server. For example, you can log off users logged on to the server and disable future logging on to the server. See [Disabling and Enabling Connections](#).

Only members of the Administrator group that you specify when you configure the application server can log out users.

When you log out users, the system does not disconnect them immediately - there can potentially be a five-minute delay before the user is logged out while the processes that the user is performing are completed.

You can control the user session timeout by changing the timeout setting in Microsoft Internet Information Services (IIS), or by changing the Web Session Timeout setting using the EPM Configurator. The default timeout setting is 20 minutes.

When you log a user off the system, the system displays a Stopped status for the user. In addition, the system notifies the user who has been logged off immediately following the first user action after logout.

To log users off an application or server:

1. Select **Navigate**, then **Administer**, and then **Consolidation Administration**.
2. Select **Administration**, then **System**, and then **Users**.
3. From the list of users, select one or more users to log out.
4. Take one of these actions:
  - Click **Logout Selected User(s)** or select **Actions**, and then **Logout Selected User(s)**.
  - Click **Logout All Users**, or select **Actions**, and then **Logout All Users**.

### Tip:

To refresh the list of users, click **Refresh** or select **Actions**, and then **Refresh**.

# Managing Servers and Applications

To manage servers and applications, you must be an Administrator.

**Related Topics:**

- [Enabling and Disabling Admin Mode](#)
- [Synchronizing Servers](#)

## Enabling and Disabling Admin Mode

If you are an administrator, you can enable Admin Mode to prevent users from logging into applications while you perform administrative maintenance tasks, such as backup and restore operations.

When you change an application to Admin Mode, all users will be logged out of the system. If there are any pending tasks (for example, consolidation, data load, or metadata load), the application will not be changed to Admin Mode and waits until the pending tasks are completed before shutting down. After the tasks are completed, you can enable Admin Mode for the application.

Users are not allowed to log into the application until you disable the Admin Mode.

To enable or disable Admin Mode:

1. Select **Navigate**, then **Administer**, and then **Consolidation Administration**.
2. Select **System**, and then **Applications**.
3. Select an application.
4. Take one of these actions:
  - From the toolbar, click the **Admin Mode** button.
  - Select **Actions**, and then **Admin Mode**.
  - Right-click on the application name and select **Admin Mode**.
5. When you are done working on the application, select the application and disable Admin Mode.

## Synchronizing Servers

The synchronization between Oracle Hyperion Financial Management application servers is based on system time. Changing the clock can affect this synchronization. For the time change to and from Daylight Saving Time (DST), Oracle recommends that you stop the servers before the time change and restart them afterward.

## Auditing Tasks

You can use the Task Audit feature to view the tasks performed by users. You can filter audited tasks by date range, application server, user, and task performed.

If you are assigned the Application Administrator role, you can view, export, and delete task audit information. If you are not an administrator, but have the View Task Audit role, you can view and export task audit information.

These user activities are logged in the task audit:

- Allocate
- AutoMatch by Account
- AutoMatch by ID
- Chart Logic (Calculate)
- Consolidation
- Create Transactions
- Data Audit Purged
- Data Clear
- Data Copy
- Data Entry
- Data Extract
- Data Load
- Data Locked
- Data Unlocked
- Delete All Transactions
- Delete Invalid Records
- Delete Transactions
- Document Attachments
- Document Detachments
- Edit Transactions
- IC Manage Periods
- IC Manage Reason Codes
- IC Manual Match
- IC Matching Report by Account
- IC Matching Report by ID
- IC Transaction Report
- Idle
- Journal Entry
- Journal Period Closed
- Journal Period Opened
- Journal Posting
- Journal Template Entry
- Journal Unposting
- Lock/Unlock Entities
- Logoff
- Logon

- Member List Extract
- Member List Load
- Member List Scan
- Metadata Extract
- Metadata Load
- Metadata Load Difference
- Metadata Scan
- Modify Application
- Post All Transactions
- Post Transactions
- Rules Extract
- Rules Load
- Rules Scan
- Security Extract
- Security Load
- Task Audit Purged
- Transactions Extract
- Transactions Load
- Transactions Scan
- Translation
- Unmatch All Transactions
- Unmatch IC Transactions
- Unpost All Transactions
- Unpost Transactions

The task audit log includes this information:

- Username
- Activity performed
- Activity start time
- Activity end time
- Server name
- Description
- Current module

The task audit log information is stored in the (*APPNAME\_TASK\_AUDIT*) table. You can back up or extract the information in the table to a file, then clear the table. You should monitor the size of the log and clear it on a regular basis.

To perform a task audit:

1. Select **Navigate**, then **Administer**, and then **Consolidation Administration**.
2. Select **Administration**, then **Audit**, and then **Tasks**.

3. **Optional:** From the **Filters** panel, select filter criteria:
  - **Application**
  - **Server**
  - **Date Range**
    - To view all dates, select **Include All**.
    - To specify a date range, select **Range**, and then specify **Minutes**, **Hours**, **Days**, or **Months**.
    - To specify specific dates, select **Custom**, and then enter a **Start** and **End** date, or click the calendar icon to select dates.
  - **Users**
  - **Tasks**
4. **Optional:** To export the audit information to a CSV file, click **Export**, or select **Actions**, and then **Export** and follow the download instructions.
5. **Optional:** To delete entries based on the current filter settings, click **Delete based on the current filter settings**, or select **Actions**, and then **Delete Filtered**.
6. **Optional:** To delete all entries from the log, click **Delete All**, or select **Actions**, and then **Delete All**.

 **Note:**

When you clear the log, a record of the clear process remains in the log and cannot be erased.

## Auditing Data

You can use the Data Audit feature to view data changes performed by users. You can filter the data changes by date range, application server, user, and dimension members.

You can use the `EnableDataAudit` metadata attribute for the accounts and scenarios for which you want to audit data changes. The audit settings for the scenario override the audit settings for the account. If the `EnableDataAudit` attribute is set to `Yes` for a scenario, all accounts in the scenario are audited, even accounts for which `EnableDataAudit` is set to `False`. If `EnableDataAudit` is set to `Override` for a scenario, all accounts for which `EnableDataAudit` is set to `True` are audited. To disable auditing of Scenario and Account members, change the `EnableDataAudit` attribute to `No`.

If you are assigned the Application Administrator role, you can view, export, and delete data audit information. If you are not an administrator, but have the View Data Audit role, you can view and export data audit information.

These user activities are logged in the data audit:

- Data Entry
- Data Clear
- Data Copy
- Data Load
- Journal Entry

The data audit log includes this information:

- User name
- Activity performed
- Time modified
- Server name
- Point of view
- Value entered for the point of view

The data audit log information is stored in the *APPNAME\_DATA\_AUDIT* table. You can back up or extract the information in the table. You should monitor the size of the log and clear it on a regular basis.

To perform a data audit:

1. Select **Navigate**, then **Administer**, and then **Consolidation Administration**.
2. Select **Administration**, then **Audit**, and then **Data**.
3. **Optional:** From the Point of View, click a dimension to select dimension members.
4. **Optional:** From the **Filters** panel, select filter criteria:
  - **Application**
  - **Server**
  - **Date Range**
    - To view all dates, select **Include All**.
    - To specify a date range, select **Range**, and then specify **Minutes**, **Hours**, **Days**, or **Months**.
    - To specify specific dates, select **Custom**, and then enter a **Start** and **End** date, or click the calendar icon to select dates.
  - **Users**
5. **Optional:** To export the audit information to a CSV file, click **Export**, or select **Actions**, and then **Export** and follow the download instructions.
6. **Optional:** To delete entries based on the current filter settings, click **Delete based on the current filter settings**, or select **Actions**, and then **Delete Filtered**.
7. **Optional:** To delete all entries from the log, click **Delete All**, or select **Actions**, and then **Delete All**.

 **Note:**

When you clear the log, a record of the clear process remains in the task audit log and cannot be erased.

## Monitoring Running Tasks

You can use the Running Tasks module to view and terminate running tasks. You can filter running tasks by application, server, user, task performed, and task status.

By default, running tasks remain in the database for 900 seconds (15 minutes).

You can change whether or not to keep the running task or purge it after the 15-minute default by modifying the AutoClearDeadTasks system setting. See [Available Configuration Settings](#).

- [Viewing Running Tasks](#)
- [Stopping Running Tasks](#)
- [Refreshing Running Tasks](#)

## Viewing Running Tasks

To view running tasks:

1. Take one of these actions:
  - From an application, select **Consolidation**, then **Maintenance**, and then **Running Tasks**.
  - Select **Navigate**, then **Administer**, then **Consolidation Administration**, then **Administration**, then **System**, and then **Running Tasks**.
2. **Optional:** From the **Filters** panel, select filter criteria:
  - **Application**
  - **Server**
  - **Users**
  - **Tasks**
  - **Status**

You can view and terminate these tasks:

- Consolidation
- Data Load
- Data Extract
- Extract Data to Database Export
- Post All Transactions
- Unpost All Transactions
- Delete All Transactions
- UnMatch All Transactions
- AutoMatch by ID
- AutoMatch by Account
- IC Matching Report
- IC Matching Report by ID
- IC Matching Report by Acct
- IC Transaction Report
- Journals Report

You can filter the running tasks by these status types:

- Initializing
- Running



- Paused
- Uninitializing
- Stopped
- Aborted
- Completed
- Not Responding
- Scheduled Stop
- Scheduled Start

## Stopping Running Tasks

Only the user who starts a task or a user assigned to the Administrator role can terminate a task that is running.

To stop a task:

1. From the Running Tasks module, select a task that you want to stop.
2. Click **Stop Tasks**, or select **Actions**, and then **Stop Tasks**.

## Refreshing Running Tasks

When you load multiple or large data or intercompany transaction files, the task may not be displayed immediately in the Running Tasks module. When you open Running Tasks, if a delay occurs in the file transfer for a data or intercompany load, the system displays a message to wait, and the screen automatically refreshes and displays the task. By default, the system automatically refreshes the Running Tasks screen in 5-second intervals and performs the automatic refresh process a maximum of 120 times. If no running task is found after 120 times, the refresh cycle stops, and you must do a manual refresh. The refresh cycle continues until all of the running tasks are completed.



### Note:

If you are the Financial Management System Administrator, you can configure Default Refresh Count and Default Refresh Interval See [Changing Configuration Settings](#).

To refresh the Running Task list:

1. From the task list, click **Refresh**, or select **Actions**, and then **Refresh**.
2. From the drop-down list, select a refresh mode:
  - **Default**
  - **Manual**
  - **5 seconds**
  - **10 seconds**
  - **15 seconds**
  - **30 seconds**

- **60 seconds**



**Note:**

The refresh mode setting is retained the next time that you log on to the application.

## Scanning For and Clearing Invalid Records

You can use the Clear Invalid Records feature to scan an application for invalid records and to remove them. You must have Administrator security access to clear invalid records.

Running the Clear Invalid Records process impacts the database, network, and Oracle Hyperion Financial Management environment and can cause performance issues. This process must be run in a maintenance window where users are not accessing the Financial Management environment.

To scan for and delete invalid records:

1. Open the application from which to delete invalid records.
2. Select **Consolidation**, then **Data**, and then **Manage**.
3. From the Manage Data page, expand **Clear Invalid Records**.



**Note:**

This feature is available only to users with Administrator security access.

4. Select an action:
  - Click **Scan** to scan for invalid records.
  - Click **Clear Invalid Records** to clear the records.

## Monitoring System Status Using HFM Insights

HFM Insights provides a dashboard for system administrators to monitor Oracle Hyperion Financial Management system status. It displays data collected from system components over a period of time and enables administrators to take corrective action if necessary.

## Accessing HFM Insights

To access HFM Insights, you must have the Oracle Hyperion Financial Management Application Administrator role.



**Note:**

The application processes are not required to be up and running for monitoring purposes.

To access HFM Insights:

1. Select **Navigate**, then **Administer**, and then **Consolidation Administration**.
2. Select **System**, and then **Insights**.

## HFM Insights Main Dashboard

When you open HFM Insights, the main page displays a table in which each row represents applications running on each server and their status. In a multi-server environment, you can filter the applications by server.

The table contains rows for each application and server combination as selected in the filter. For example, if you selected three applications and two servers, the table displays six rows: Application 1 on Server 1, Application 2 on Server 2, and so on. For each row, the following information is displayed:

- Application icon, application name, and server name
- Chart based on the following Key Performance Indicators (KPI). The data is limited to the last two weeks:
  - **Users** - number of users during the past two weeks
  - **Errors** - number of errors during the past two weeks
  - **Tasks** - number of tasks during the past two weeks
  - **Memory** - Physical memory in MB
  - **CPU** - CPU usage
- Status Overview:
  - **Status icon** displaying one of the following statuses:
    - \* Admin Mode
    - \* Crash
    - \* Down (process is not loaded)
    - \* Unresponsive
    - \* Up (process is up and running)
  - **Start Time** - the last time the XFMDatasource process was started
  - **Up Time** - Uptime calculation, which is the difference between the Start Time and the last ping time
  - **Last Health Check** - Timestamp of last successful ping to XMFDataSource process from HsxServer log

By default, all metrics are displayed. You can deselect any of the metrics from an individual graph. For example, you can select only CPU and deselect the other metrics. The graph will scale according to your selections.

You can use the Refresh option at any time to refresh the screen.

## Filtering Applications in HFM Insights

**Application filter** - lists all the applications for which you are an Application Administrator. You can select one or more applications. By default, the first five applications in the list are selected and the table displays the status of those applications.

**Server filter** - lists all the available servers. You can select one or more servers. If there is only one server in the system, the server filter is not displayed. If the server filter is displayed, by default the first server is selected and the table lists the application status on that server.

**Show filter** - enables you to select the metrics to be displayed in the graph. It is a multi-selection list. By default, all the metrics are selected and the graph displays all the metrics: Users, Errors, Tasks, Memory, and CPU.

## Application Details in HFM Insights

When you click an application icon, a new tab opens, which displays additional detailed metrics for the application running on that server.

The tab title is <Application Name>@<ServerName>.

- **Show** - Filter for time periods to display. By default, data is displayed for the last 14 days.
- **Refresh**- Select Manual or Auto Refresh. The default value is Manual.

The Application page displays the following sections:



### Application Summary

- Application icon, application name, and server name
- Status Overview, as shown on the main page
- **Application Dimensions** - Application Dimensions and total members in each dimension
- **Restart** - Restarts the XFMDatasource process

### Statistics

The Statistics section displays detailed data statistics based on a selected Scenario and Year combination. You can choose when to manually update the statistics.

To update Statistics:

1. From the **Statistics** section, click the **Data** tab.
2. Click the **Member Selector** , select a Scenario and Year, and click **OK**.
3. Click the **Execute** button  to run the process and return statistics on the selected Scenario and Year combination.

#### Note:

When you click the **Execute** button and update statistics, the button is disabled for a certain period of time while the process runs. The task runs in the background, and you can navigate to other parts of the application while it is running. When it is done, the button is re-enabled and the table is refreshed with the updated statistics. You can also click **Refresh** to see the updated values.

If there are no records in a particular table, the system does not display any statistics for that table.

### Data tab

- Data (DCE and DCN total record count per Scenario/Year combination)
- Invalid records (invalid member ID), DCE and DCN only, Total count only (not per entity)
- Near Zero Value Records for all periods
- Near Zero Value Calculated Records for all periods
- PFlow History (per Scenario/Year)
- Input Data (if any period in the year is Input or Line Item Details)

### Table Count tab

The Table Count tab displays record counts of the following Oracle Hyperion Financial Management system tables and the times they were last executed.

- Task Audit
- Data Audit
- System Messages (Error Log table)
- Rules Profile data

### Performance Overview

The Performance section displays a chart based on the same KPI as on the main screen.

### User

**Frequent users** - Bar graph displaying the top 10 users with the maximum number of sessions over a time period, based on the Task Audit data. By default, data is displayed for the last 14 days, but you can change the time period using the Show filter.

**Session Details** - Details of user sessions, including User name, Time Started, Time Ended, and Duration. You can sort any of the columns.

You can filter by User. In the User Search box, type a full or partial user name.

In the Duration Search box, to view only users logged in for a specific duration, enter a time value. For example, to view users logged in for 1 hour, enter 1h. You can use the percent sign (%) as a wildcard. For example, to view users with a duration less than one hour, in the Search box, enter %h.

### Memory

The Memory section displays a graph of system memory parameters, based on System Messages data. It includes:

- Total Memory
- Used Memory
- NumCubesInRAM
- NumDataRecordsInRAM
- NumRecordsInLargestCube
- MinDataCacheSizeInMB
- MaxDataCacheSizeInMB
- MaxNumCubesInRAM

For details on memory parameters, see [Available Configuration Settings](#).

## Tasks

This section contains four tabs that display Task information.

### Tasks View

The Tasks view displays the top 10 frequently executed tasks in the selected time span, based on Task Audit data. For example, you can quickly view the number of Logons, or Consolidations that occurred during the last two weeks.

- **Metric** - filter for frequency time metric:
  - Frequency - Count of tasks that occurred in the selected time span
  - Total Time - Total time taken by the tasks that occurred in the selected time span
  - Average Time - Average time taken by the tasks that occurred in the selected time span
- **User** - filter by User who performed the task. In the User Search box, type a full or partial user name, or use the percent sign (%) as a wildcard.

### Users View

The Users View tab contains a bar chart showing the top 10 users who performed a particular task.

- **Metric** - filter for task times:
  - Frequency - Count of tasks performed by the users in the selected time span
  - Total Time - Total time taken by the tasks performed in the selected time span
  - Average Time - Average time taken by the tasks performed in the selected time span
- **Tasks** - filter by Task. By default, Consolidation is selected.

### Months View

The Months View displays a bar chart showing a particular task on a monthly basis. You can compare task activity by month.

- **Metric** - filter for task times:
  - Frequency - Count of tasks performed for a selected month or months
  - Total Time - Total time taken by the tasks for the selected month or months
  - Average Time - Average time taken by the tasks for the selected month or months
- **Tasks** - filter by Task. By default, Consolidation is selected.

### Task Details

This tab displays detailed information about each task, based on Task Audit data. It includes the User name, Activity name, Time Started, Time Ended, Duration, and Description. You can sort any of the columns.

You can filter by User, Activity, Duration, and Description.

### Errors

The Errors section displays the top 10 frequent errors, based on System Messages data.

- **Error Frequency** - Displays the Error Code. You can hover over the error code to view its description.
- **Error Details** - Displays the Error Code, System Message Summary, Date and Time of the error for the selected application on the selected server for a particular span of time. You can sort on Error Code and Date-Time. You can filter by Error Code and System Message Summary.

# 3

## Managing Application Security

Security and access rights enable you to control access to Oracle Hyperion Financial Management applications and application elements. Setting up security enables you to protect data and prevent unauthorized users from changing data. For example, you can restrict access to certain data elements or forms within an application.

Security exists at two levels:

- Authentication by an external provider
- Financial Management security, in which users and groups are assigned to applications and application elements are assigned to security classes

There are two ways to set up security for Financial Management applications:

- Load a security file into an application. See [Loading Application Security](#).
- Use the Oracle Hyperion Shared Services Console to set up security information. See [Launching the Shared Services Console from Financial Management](#).

These security roles are required for application administration. For information on assigning roles, see the *Oracle Enterprise Performance Management System User Security Administration Guide*.

Security Role	Description
Financial Management Administrator	Required for accessing administration functionality.
Dimension Editor	Creates and manages import profiles for dimension creation, as well as creating and managing dimensions manually. Required to access Classic application administration options.
Financial Management Application Creator/ Financial Management Manager	Creates applications. Users with this role can create applications, but can change only the dimensions to which they have access permissions. The Financial Management Application Creator can create Consolidation applications and Generic applications. To create applications, the user must also be a member of the Application Creators group specified in the Configuration Utility.

## Application Security Considerations

Oracle Hyperion Financial Management security offers flexibility in securing application elements and tasks. Because security classes are assigned to application elements as they are created, you should design your security system before you set up your applications.

After you design a security system for one application, you can extract the security elements for backup or loading into another application. See [Loading Application Security](#) and [Extracting Application Security](#).



Before setting up security in Financial Management, you should consider these questions:

- How do you want to group and classify Financial Management tasks and application elements?
- How do you want to group users?
- What level of access right should be assigned for your users and groups?
- What security classes do you want to assign to application elements as they are created?

## Launching the Shared Services Console from Financial Management

Before you can set up security for Oracle Hyperion Financial Management applications, you must do these tasks:

1. Create Financial Management applications.
2. Provision users by assigning users and groups to applications and assigning user roles. See the *Oracle Enterprise Performance Management System User Security Administration Guide*.

You can then use the Oracle Hyperion Shared Services Console to set up security for Financial Management applications. In the console, you can do these application tasks:

- Assign users and groups
- Assign user permissions to security classes
- Run security reports

To launch the Shared Services Console:

1. From Financial Management, select **Navigate**, then **Administer**.
2. Select **Shared Services Console**.

## Selecting Users and Groups for Assigning Security Classes

Only a user assigned to the Provisioning Manager role can define users and groups. See the *Oracle Enterprise Performance Management System User Security Administration Guide*.



### Note:

In Hyperion Financial Management, Users and Groups cannot have the same names.

By default, the Access Control page displays the first 100 provisioned groups and users. Groups are displayed first, followed by users in the table. Users and Groups can be distinguished by their icons in the table.

To select users and groups for an application:

1. From the Oracle Hyperion Shared Services Console, expand **Application Groups**, right-click the application name, and select **Assign Access Control**.
2. Select **Users/Groups**, or select **Actions**, and then **Users/Groups**.

3. Select an option: **Users** or **Groups**.
4. From **Available Users**, or **Available Groups**, select users and groups to assign to the application. and use the arrow keys to move them to the **Selected Users** or **Selected Groups** column.
5. Click **OK**.

## Setting Up Security Classes

Security classes determine user and group access rights to application elements.

You can perform these procedures:

- [Creating Security Classes](#)
- [Deleting Security Classes](#)
- [Selecting Security Classes](#)



### Note:

Only users assigned to the Provisioning Manager role can define security classes for applications.

After you define security classes for an application, you can assign the security classes to application elements such as accounts and entities.

A user's or group's ability to access application elements depends on the security classes to which the user or group belongs and on the security class associated with the application elements.

A system-generated security class called [Default] is created as part of an application. It cannot be deleted or modified by users. Access rights can be assigned to the [Default] security class. Any member that is not assigned a security class is treated as if it has the Default security class.

## Creating Security Classes

To create security classes:

1. From the Oracle Hyperion Shared Services Console, expand **Application Groups**, right-click the application name, and select **Assign Access Control**.
2. Select **Add Security Class**, or select **Actions**, and then **Add Security Class**.
3. For **Name**, enter a name for the security class.

The name must be unique and can contain a maximum of 80 characters. It can include spaces, but it cannot start with a space. These characters are not allowed:

- Asterisk (\*)
- At sign (@)
- Comma (,)
- Curly brackets ({})
- Double quotation marks""

- Minus sign (-)
  - Number sign (#)
  - Period (.)
  - Plus sign (+)
  - Semicolon (;)
  - Slash mark (/)
4. Click **OK**.

## Deleting Security Classes

When you no longer need a security class, you can delete it from the security class dimension. Before you delete a security class from an application, you must disassociate it from the application elements to which it is assigned.

You can disassociate an entity, account, or scenario from a security class by modifying the security class in the metadata file. You can disassociate a journal from a security class by modifying the journal file or by updating the security class for the journal in the Process Journals module.

To delete security classes:

1. From the Oracle Hyperion Shared Services Console, expand **Application Groups**, right-click the application name, and select **Assign Access Control**.
2. From the list of security classes, select the security class row/column header based on its view in the table.



### Tip:

To delete multiple security classes, use the Ctrl key and select row/ column headers.

3. Select **Delete Security Class**, or select **Actions**, and then **Delete Security Class**.
4. Click **Yes** to confirm deletion.

## Selecting Security Classes

By default, the Security Classes page displays the available security classes, and the system displays the first 100 security classes.

To select security classes for an application:

1. From the Oracle Hyperion Shared Services Console, expand **Application Groups**, right-click the application name, and select **Assign Access Control**.
2. Select **Security Classes**, or select **Actions**, and then **Security Classes**.
3. From **Available Security Classes**, select the security classes to assign to the application, and use the arrow key to move them to the **Selected Security Classes** column.
4. Click **OK**.

## Assigning User Access to Security Classes

After you define users and groups and create security classes, you can specify the type of access that each user and group has to each security class in the application.

You can assign users one of five access types: None, Metadata, Read, Promote, or All. You can use the Pivot feature to toggle between two views for the assign access table. For example, if users and groups are on the rows and security classes are on the columns and you click Pivot, the users and groups move to the columns and security classes move to the rows.

When you grant users access to a security class, you can enable email alerts, which can be used for intercompany reporting, and also to inform users of process unit status changes for entities and scenarios that use the security class. To receive email alerts for process control, a user must have All or Promote access to the security class.



### Note:

A user assigned to the Application Administrator role for an application has access to all information in the application.

**Table 3-1 User Access Level**

Access Level	Description
None	No access to elements assigned to the security class.
Metadata	View a specified member in a list but cannot view or modify data for the member.
Read	View data for elements assigned to the security class but cannot promote or reject.
Promote	View data for elements assigned to the security class and can promote or reject.
All	Modify data for elements assigned to the security class and can promote and reject.

To assign user access to security classes:

1. From the Oracle Hyperion Shared Services Console, expand **Application Groups**, right-click the application name, and select **Assign Access Control**.
2. Select cells for which to assign access rights.



### Tip:

Use the Shift and Ctrl keys to select multiple cells. Select a column or row by clicking in the column or row header. To change the display of columns and rows, click **Pivot**.

3. Right-click and select the access level to assign.

 **Note:**

See [Table 1](#).

- **None**
  - **Metadata**
  - **Read**
  - **Promote**
  - **All**
4. **Optional:** To add an email alert, select cells in the table, right-click and select **Enable Email Alert**.

 **Caution:**

The alerting process uses the email addresses stored in the authentication files, such as MSAD, LDAP, or Native Directory.

 **Note:**

To remove email alerts, select the cell and click **Disable Email Alert**.

5. Click **Save**.

## Setting Up Email Alerting

You can use email alerting for intercompany transactions and during the process management review process. Email alerts help highlight a key event or data change in the system. For example, you can send an email alert that an intercompany transaction is mismatched and needs to be matched, or that a process unit is ready for the next promotion level.

 **Note:**

The alerting process uses the email addresses that are stored in your external authentication provider, such as LDAP, MSAD, or Native Directory.

Users with the Application Administration role do not automatically receive email alerts. For a user with the Application Administrator role to receive email alerts, set up the user as a separate user and assign the security role to receive alerts.

## Running Security Reports

You can run security reports on the information that you selected while setting up security for the application. You can run reports for classes by user, roles by user, classes and roles by user, and users by group. You can view the report online or you can export it to a file.

To create a security report:

1. From the Oracle Hyperion Shared Services Console, expand **Application Groups**, right-click the application name, and select **Assign Access Control**.
2. Select **Security Reports**, or select **Actions**, and then **Security Reports**.
3. Select a report option:
  - **Access Rights** and select options:
    - **Classes by User**
    - **Roles by User**
  - **Users by Group**
4. Select a **Report Format**: PDF, RTF, HTML, XML, XSLX.

 **Note:**

The HFM-Format is not supported for security reports.

5. **Optional:** Select a **Template**.
6. Select an option:
  - **Launch Report** to open the report in a new page.
  - **Export to File** to save the report in the selected report file format.

## Loading Application Security

 **Caution:**

You can only load security classes for Classic Oracle Hyperion Financial Management applications.

You must load application security before you can load other information to an application. If you are loading several application elements at once, the system loads the security files first.

Security information load files can be in an ASCII or Unicode format. The default file name extension for security information load files is SEC.

You can load users, security classes, role access, and security class access.

To remove a role from a user or group, you must modify the role in the Shared Services Console. See the *Oracle Enterprise Performance Management System User Security Administration Guide*.

 **Note:**

Oracle recommends that you add Financial Management to the exceptions for your Web pop-up blocker. When you perform some tasks such as loading data, a status window pops up showing the task status. If you have a pop-up blocker enabled on your computer, the status window is not displayed.

To load application security:

1. Open the application.
2. Select **Consolidation**, then **Load**, and then **Application Elements**.
3. In the **Security** section, enter the file name to load, or click **Browse** to find the file.

 **Note:**

By default, application security information files use the SEC file extension. The load process accepts other file extensions such as TXT or CSV, however, Oracle recommends that you use the SEC file extension.

4. **Optional:** Select **Clear All** to clear security information for the application before loading the new security information.

 **Caution:**

You can use the Clear All option only if you have been assigned the Application Administrator and Provisioning Manager roles. Also, if you use this option, you will have to reprovision users, as all users (including the user doing the clear) will be removed in this process. For information on provisioning users, see the *Oracle Enterprise Performance Management System User Security Administration Guide*. Before selecting the clear option, review [Clearing and Loading Security Information](#).

5. From **Delimiter**, select the character used to separate information in the file. These characters are valid:

, ~ @ \$ % ^ & | : ; ? \

 **Note:**

You must use a character that is not used in the file name or in any other way in the file. For example, if you use the comma in an entity description, you cannot use the comma as the delimiter.

6. From the **Filters** section, select the types of security information to load.



**Tip:**

To reset the filter selections, click **Reset**.

7. Click **Load**.
8. **Optional:** To download the log file, click **Download Log**. Click **Open** to display the log file, or **Save** and select a location to save the file locally.

## Clearing and Loading Security Information

Before you begin a security file load, you can clear security information from an application and then load the new security information. For example, if you plan to change security class Class1 to Class2 during the security load, you must make the change to all application elements that reference the Class1 security class.

Because the system generates new security references for application elements that use security class information, you must perform prerequisite steps before you load the new security information, and perform follow-up steps after you load it.

To clear security information and load a new security file:

1. Extract application elements from the application. See [Before Clearing Security Information](#).
2. Select to clear existing security information and load a new security file.
3. Load application elements to the application. See [After Clearing Security Information](#).



**Note:**

You must be assigned to the Application Administrator security role to be able to perform these procedures.

## Before Clearing Security Information

Before you clear security information and load a security file, you must perform these tasks for the specified application elements that utilize security class information.

### Metadata

To update metadata before clearing and loading security information:

1. Extract all application metadata elements in the application.
2. Make changes to the security class information of the metadata elements as necessary.

### Journals

To update journals before clearing and loading security information:

1. Unpost posted journals in the application.
2. Reject approved journals so that the journal status reverts to Working.
3. Extract all journals.



4. Make changes to the journal security class information as necessary.

### **Grids**

To update grids before clearing and loading security information:

1. Extract all grids that have a security class assigned.
2. Make changes to the grid security class information as necessary.

### **Data Forms**

To update data forms before clearing and loading security information:

1. Extract all data forms that have a security class assigned.
2. Make changes to the data forms security class information as necessary.

## After Clearing Security Information

After you clear security information and load a security file, you must perform these tasks for the specified application elements that utilize security class information.

### **Metadata**

To update metadata:

1. Make sure the metadata information is cleared.
2. Load the updated metadata file to the application.

### **Journals**

To update journals after clearing and loading security information:

1. Load the updated journal file.
2. Post journals that you unposted before clearing and loading security information.
3. Approve journals that you rejected before clearing and loading security information.

### **Grids**

To update grids after clearing and loading security information:

1. Load the updated grid files.
2. Select the option to overwrite existing documents.

### **Data Forms**

To update data forms after clearing and loading security information:

1. Load the updated data form file.
2. Select the option to overwrite existing documents.

### **Folders**

To update folders after clearing and loading security information:

1. Delete folders that may have an incorrect security class assigned.
2. Add new folders to the application.

## Reports

To update reports after clearing and loading security information:

1. Reload all reports with an assigned security class.
2. Provide the new security class assignment if applicable.

## Task Lists

To update task lists after clearing and loading security information:

1. Reload all task lists with an assigned security class.
2. Provide the new security class assignment if applicable.

# Extracting Application Security

You can extract application security to view or modify it in a text editor. When you extract application security from an application, save the file in a format that supports multibyte character sets (MBCS). By default, application security files use the SEC file extension.

You can extract these types of security information:

- Users and groups
- Security classes
- Role access
- Security class access

### Note:

Oracle recommends that you periodically extract security to a backup file. For information on backing up security information, see the *Oracle Enterprise Performance Management System Installation and Configuration Guide*.

To extract application security:

1. Open the application.
2. Select **Consolidation**, then **Extract**, and then **Application Elements**.
3. In the **Security** section, from **Delimiter**, select the character used to separate information in the file.

These characters are valid:

, ~ @ \$ % ^ & | : ; ? \

### Note:

You must use a character that is not used in the file name or in any other way in the file. For example, if you use the comma in an entity description, you cannot use the comma as the delimiter.

4. From **Filters**, select the types of security to extract.



**Tip:**

To reset the selections, click **Reset**.

5. Click **Extract**.
6. Follow the download instructions displayed in the browser to download the extracted file.  
The instructions vary depending on the Web browser that you are using. Make sure to save the file in the Web directory that you set up.
7. **Optional:** To download the log file, click **Download Log**. Click **Open** to display the log file, or **Save** and select a location to save the file locally.

# 4

## Managing Metadata

Metadata is defined as the structural elements of an application that describe and store data, for example, dimension names, member names, properties, exchange rates, and security.

To define metadata, create an XML or APP metadata file and load it to an application.

Oracle Hyperion Financial Management allows only ASCII characters in metadata.



### Note:

You must set up security for an application before you can load metadata.

Sample metadata files are included when you install Sample Applications for Financial Management. The files are located in the Sample Applications folder in the directory to which you installed Financial Management.

## Defining Accounts

The Account dimension defines the chart of accounts for an application. You define accounts with the attributes in the Account Member Attributes table.

Table 4-1 Account Member Attributes

Attribute	Description
AccountType	<p>(Required) One of these values:</p> <ul style="list-style-type: none"> <li>• ASSET—Store values that represent the assets of a company</li> <li>• LIABILITY—Store point-in-time balances that represent the liabilities of a company</li> <li>• REVENUE—Store periodic and year-to-date values that increase net worth if the value is positive</li> </ul> <p><b>Note:</b> In Oracle Hyperion Financial Management releases prior to 4.1, this account type was called Income.</p> <ul style="list-style-type: none"> <li>• EXPENSE—Store periodic and year-to-date values that decrease net worth if the value is positive</li> <li>• FLOW—Store periodic and year-to-date values</li> <li>• BALANCE— Store unsigned values that relate to a particular point in time</li> <li>• BALANCERECURRING—Store unsigned values that relate to a particular point in time and that re-occur in future periods</li> <li>• CURRENCYRATE—Store currency rate information</li> <li>• GROUPLABEL—Use the account for grouping purposes</li> <li>• DYNAMIC—Indicates that the account value is calculated dynamically from the data that you are viewing</li> </ul>
CalcAttribute	<p>Description of the calculations performed in the rules file for this account</p> <p>This information is displayed as part of cell information in data forms and data grids. It can contain up to 80 characters, including spaces.</p>
CustomTop	<p>Which TopMember in the hierarchy of a Custom dimension is valid for the account</p> <p>Only the specified member, including all descendants, is valid for the account.</p> <p>The number of attributes is based on the number of Custom dimensions defined for the application. The attribute name changes to reflect the Custom dimension alias. For example, Custom in the attribute name is replaced with the Custom dimension alias.</p>
DefaultParent	<p>The default parent for the account</p>
Description	<p>The account description</p> <p>The description can contain up to 80 characters, including spaces, and cannot use an ampersand (&amp;) or backslash (\).</p>

Table 4-1 (Cont.) Account Member Attributes

Attribute	Description
EnableCustomAggr	<p>Whether Custom dimension data is aggregated for the current account</p> <p>This attribute is used for special totals, not summing. Specify Y if the account can aggregate with Custom dimensions or N if it cannot.</p> <p>The number of attributes is based on the number of Custom dimensions defined for the application. The attribute name changes to reflect the Custom dimension alias. For example, Custom in the attribute name is replaced with the Custom dimension alias.</p>
EnableDataAudit	<p>Whether the account can be audited</p> <p>Specify Y to enable account auditing or N to disable auditing. The default is N. This attribute, when applied to an account or scenario, determines what can be audited.</p>
ICPTopMember	<p>The Intercompany PartnerTopMember for the account</p> <p>The specified member and all its descendants are valid for the account. All other members of the Entity dimension are not valid for the account.</p>
IsCalculated	<p>Whether the account is calculated</p> <p>Only base-level accounts can be calculated. If a base-level account is calculated, you cannot manually enter values. Specify Y if the account is to be calculated; otherwise, specify N.</p>
IsConsolidated	<p>Whether values for the account are consolidated to parent entities. If the account is not consolidated, it is ignored during consolidation. Specify Y if the account is to be consolidated when consolidation is performed or N if the account is not to be consolidated.</p>
IsICP	<p>Specifies whether the account is an intercompany account.</p> <ul style="list-style-type: none"> <li>• Y if ICP transactions, including self-ICP transactions, are allowed for the account</li> <li>• N if ICP transactions are not allowed for the account</li> <li>• R if ICP transactions are allowed for the account, but the account is restricted from ICP transactions with itself</li> </ul>

Table 4-1 (Cont.) Account Member Attributes

Attribute	Description
Member	<p>Specifies the name for the account. This attribute is required. The name must be unique. It can contain up to 80 characters, including spaces, but cannot start with a space.</p> <p>Do not use these characters in an account name:</p> <ul style="list-style-type: none"> <li>• Asterisk (*)</li> <li>• At sign (@)</li> <li>• Comma (,)</li> <li>• Curly brackets ({})</li> <li>• Double quotation marks""</li> <li>• Greater than symbol (&gt;)</li> <li>• Less than symbol (&lt;)</li> <li>• Minus sign (-)</li> <li>• Number sign (#)</li> <li>• Period (.)</li> <li>• Plus sign (+)</li> <li>• Semicolon (;)</li> <li>• Slash mark (/)</li> </ul>
NumDecimalPlaces	<p>Specifies the number of digits to the right of the decimal point to be displayed for account values. This attribute is required. Specify a value from 0 to 9.</p>
PlugAcct	<p>Specifies the account name used for identifying discrepancies in intercompany transactions. The PlugAcct attribute is required when the IsICP attribute for the account is selected. It must be blank or the name of a valid account. If blank, intercompany eliminations for the account are not processed.</p>
SecurityClass	<p>Specifies the security class that defines the users who can access the account data. Security class names can contain up to 80 characters. Security access applies only to account data.</p>
Submission Group	<p>Specifies the submission group for applications that use phased submission.</p> <p>Enter a number from 1 to 9 to specify a submission group, or zero to exclude the account from process control. The default value is 1.</p>
UserDefined1, UserDefined2, UserDefined3	<p>Stores custom information for the account. You can enter a maximum of 256 characters. The UserDefined1, UserDefined2, and UserDefined3 functions retrieve the text stored in this attribute.</p>

Table 4-1 (Cont.) Account Member Attributes

Attribute	Description
UsesLineItems	<p>Specifies whether an account can have line items. If selected, the account uses line items in scenarios for which line items are enabled. Specify Y if the account uses line items or N if the account does not use line items.</p> <p><b>CAUTION:</b> If you change this attribute after line-item detail is entered, the stored line-item detail may no longer be valid for the account. These behaviors occur:</p> <ul style="list-style-type: none"> <li>• If the account accepted line items and now it cannot, the line-item detail stored in the database is no longer valid. Only the total is displayed.</li> <li>• If the account did not accept line items and now it can, there is a total amount but no corresponding line-item detail information for the account. You can extract the total and then load it as line-item detail data so that the total matches the line-item detail information.</li> </ul>
XBRL Tags	Specifies XBRL tags for the account. You can enter a maximum of 225 characters.

## Account Type Behavior

Each account has an account type. Account types determine how child accounts are aggregated to parent accounts and how account balances accumulate over time. When data is input to base-level accounts, results are automatically rolled up through the hierarchy.

Account types determine whether child values are added to or subtracted from their parent value. This determination enables you to build financial calculations directly into the chart of accounts.

### Note:

When you load data, Oracle Hyperion Financial Management assumes that the natural sign for the data is positive.

For example, the ASSET account type does not total across periods. If you debit an ASSET account, the value that you enter is added to the account. If you credit it, the value is subtracted. The default translation for this account type is the value in the DefaultRateforBalance Accounts field.

A REVENUE account provides a year-to-date total. The DYNAMIC account type is needed for correct calculation of parent values for Custom dimensions, time periods, and period-to-date views. You can use the GROUPLABEL account type to group related accounts that do not need to be aggregated to a total. For example, you can create a top-level account named Balance Sheet Accounts that groups balance sheet accounts. All account types, except for GROUPLABEL, store data.



**Note:**

Scaling does not apply for FLOW, BALANCE, or DYNAMIC account types.

**Table 4-2 Account Type Behaviors**

Type	YTD Total	Debit	Credit	Default Translation
ASSET	No	+	- <sup>**</sup>	DefaultRateForBalance Accounts
LIABILITY	No	-	+	DefaultRateForBalance Accounts
REVENUE	Yes	-	+	DefaultRateForFlow Accounts
EXPENSE	Yes	+	-	DefaultRateForFlow Accounts
FLOW	Yes	+	-	None
BALANCE	No	+	-	None
BALANCE RECURRING	No	+	-	None
CURRENCYRATE	No	N/A	N/A	N/A
GROUPLABEL	N/A	N/A	N/A	N/A
DYNAMIC	N/A	N/A	N/A	N/A

**Note:**

- <sup>\*</sup> - This symbol indicates Addition.
- <sup>\*\*</sup> - This symbol indicates Subtraction.

The following table indicates how an account type behaves when totaled into a specific type of parent account. The columns represent the account type of the parent accounts. For example, when aggregated, ASSET account values are added into parent ASSET and EXPENSE accounts and subtracted from parent LIABILITY and REVENUE accounts.

**Note:**

The abbreviations represent the first one or two letters of the account types. A No displayed in the column indicates that the account type is not aggregated into the parent account.

**Table 4-3 Account Type Behaviors During Aggregation into Parent Accounts**

Type	Parent Account									
	A	L	R	E	F	B	BR	C	G	D
ASSET	+	- <sup>**</sup>	-	+	+	+	+	No	No	No
LIABILITY	-	+	+	-	+	+	+	No	No	No
REVENUE	-	+	+	-	+	+	+	No	No	No
EXPENSE	+	-	-	+	+	+	+	No	No	No
FLOW	+	+	+	+	+	+	+	No	No	No
BALANCE	+	+	+	+	+	+	+	No	No	No
BALANCE RECURRING	+	+	+	+	+	+	+	No	No	No
CURRENT RATE	No	No	No	No	No	No	No	No	No	No
GROUP LABEL	No	No	No	No	No	No	No	No	No	No
DYNAMIC	No	No	No	No	No	No	No	No	No	No



**Note:**

- <sup>\*</sup> - This symbol indicates Addition.
- <sup>\*\*</sup> - This symbol indicates Subtraction.

This example illustrates how account types are aggregated into parent accounts:

```

Total Assets 80
├─ Fixed Assets 100
└─ Amortization 20
  
```

In this example, Total Assets is an ASSET account and the parent of Fixed Assets (an ASSET account) and Amortization (a LIABILITY account). When the accounts are aggregated into the parent account, the Fixed Assets value of 100 is added, the Amortization value of 20 is subtracted, and the resulting value for Total Assets is 80.

## Defining Dynamic Accounts

Dynamic accounts are accounts with values that are dynamically calculated when the data is requested. The values for dynamic accounts are not stored. The most common type of dynamic calculation is ratio calculation.

To define a dynamic account and calculation:

1. Set up an account that uses the Dynamic account type.

Only base accounts can be dynamic.

### Note:

These account attributes are ignored for dynamic accounts: IsCalculated, IsConsolidated, EnableCustomAggr, UsesLineItems.

2. In a rules file, create a Sub Dynamic () section.
3. In the rules file, define a calculation.

For more information on writing calculations, use the guidelines for creating rules.

## Defining Custom Members

Custom dimensions are associated with the Account dimension and provide additional detail for accounts. You define Custom members by using the attributes in the Custom Member Attributes table.

**Table 4-4 Custom Member Attributes**

Attribute	Description
DefaultParent	Specifies the default parent for the Custom dimension member.
Description	Specifies the description for the Custom member. The description can contain up to 80 characters, including spaces.
IsCalculated	Specifies whether the base-level Custom account is calculated. If a base-level Custom account is calculated, you cannot manually enter values. Specify Y if the Custom account is to be calculated or N if the Custom account is not to be calculated.

Table 4-4 (Cont.) Custom Member Attributes

Attribute	Description
Member	<p>Specifies the name for the Custom member. This attribute is required. The name must be unique and can contain up to 80 characters, including spaces, but cannot start with a space.</p> <p>The name of a Custom dimension member cannot duplicate the name of a consolidation method.</p> <p>Do not use these characters in the Custom member name:</p> <ul style="list-style-type: none"> <li>• Asterisk (*)</li> <li>• At sign (@)</li> <li>• Comma (,)</li> <li>• Curly brackets ({} )</li> <li>• Double quotation marks ""</li> <li>• Minus sign (-)</li> <li>• Number sign (#)</li> <li>• Period (.)</li> <li>• Plus sign (+)</li> <li>• Semicolon (;)</li> <li>• Slash mark (/)</li> </ul>
SecurityClass	<p>Specifies the security class name that defines the users who can access the Custom dimension data. Security class names can contain up to 80 characters. Security access applies only to data.</p>
Submission Group	<p>Specifies the submission group. The value can be a number from 0 to 99.</p> <p>The default is blank. A blank value defaults to the value of 1.</p> <p>If you set the submission group to zero (0), the account is not included in the review process.</p>
SwitchSignForFlow	<p>Specifies sign change (Debit or Credit) for FLOW accounts that use these rules:</p> <ul style="list-style-type: none"> <li>• ASSET to LIABILITY</li> <li>• LIABILITY to ASSET</li> <li>• EXPENSE to REVENUE</li> <li>• REVENUE to EXPENSE</li> <li>• BALANCE to FLOW</li> <li>• FLOW to BALANCE</li> </ul> <p>Specify Y if the sign for the account is switched or N if the sign for the account is not switched.</p>
SwitchTypeForFlow	<p>Specifies the account type change for FLOW accounts that use these rules:</p> <ul style="list-style-type: none"> <li>• ASSET to EXPENSE</li> <li>• EXPENSE to ASSET</li> <li>• LIABILITY to REVENUE</li> <li>• REVENUE to LIABILITY</li> <li>• BALANCE to FLOW</li> <li>• FLOW to BALANCE</li> </ul> <p>Specify Y if the account type for the account is switched or N if the account type for the account is not switched.</p>

**Table 4-4 (Cont.) Custom Member Attributes**

Attribute	Description
UserDefined1, UserDefined2, UserDefined3	Stores custom information for the dimension member. You can enter a maximum of 256 characters. The UserDefined1, UserDefined2, and UserDefined3 functions retrieve the text stored in this attribute.

## Defining Entity Members

Entities represent the organizational structure of the company, such as divisions, subsidiaries, plants, regions, countries, legal entities, business units, departments, or any organizational unit. They define the consolidation path for data. When you run a consolidation for the Entity dimension, data rolls up from children to parents as defined in the dimension hierarchy. You can create multiple consolidation paths by placing an entity child member under more than one parent. You define entity members by using the attributes in the Entity Member Attributes table.

**Table 4-5 Entity Member Attributes**

Attribute	Description
AllowAdjFromChildren	Specifies whether journal postings from children are permitted for the parent entity. For entities that roll up to more than one parent, you can enable this attribute for the parent entity. Specify Y if journal postings from children are permitted or N if journal postings from children are not permitted.
AllowAdjs	Specifies whether journal postings are permitted for this entity. Specify Y if journal postings are permitted for the entity or N if journal postings are not permitted for the entity.
DefaultParent	Specifies the default parent for the entity.
DefCurrency	Specifies the default currency for the entity. This attribute is required.
Description	Specifies the description for the entity. The description can contain up to 80 characters, including spaces.
HoldingCompany	Specifies the holding company for the entity, which identifies the owner of an entity member. Can be the name of an entity or blank.
IsICP	Specifies whether entities can be partners in intercompany transactions. Specify Y if the entity is an intercompany entity or N if the entity is not an intercompany entity. A member for which you select ICP is automatically displayed as a member in the ICP dimension.

Table 4-5 (Cont.) Entity Member Attributes

Attribute	Description
Member	<p>Specifies the name for the entity. This attribute is required. The name must be unique and can contain up to 80 characters including spaces but cannot start with a space.</p> <p>Do not use these characters in the entity name:</p> <ul style="list-style-type: none"> <li>• Asterisk (*)</li> <li>• At sign (@)</li> <li>• Comma (,)</li> <li>• Curly brackets ({ })</li> <li>• Double quotation marks ""</li> <li>• Minus sign (-)</li> <li>• Number sign (#)</li> <li>• Period (.)</li> <li>• Plus sign (+)</li> <li>• Semicolon (;)</li> <li>• Slash mark (/)</li> </ul> <p>You cannot use ALL as the name of an entity.</p>
SecurityAsPartner	<p>This attribute enables you to specify a security class for an entity acting as an intercompany partner. Specifies the name of a valid security class for the ICP entity.</p>
SecurityClass	<p>Specifies the name of a valid security class of users who can access the data of the entity. Security class names can contain up to 80 characters.</p>
UserDefined1, UserDefined2, UserDefined3	<p>Stores custom information for the entity. You can enter a maximum of 256 characters. The UserDefined1, UserDefined2, and UserDefined3 functions retrieve the text stored in this attribute.</p>

## Defining Scenario Members

The Scenario dimension represents a set of related data, such as budget, actual, or forecast. You define scenario members by using the attributes in the Scenario Member Attributes table.

The frequency of a scenario specifies the time period level at which data can be input. You can input and view either periodic or year-to-date data values. For example, if you input data as year-to-date values, when you select Periodic as the data view, the system automatically derives the periodic values from the year-to-date values.

For each scenario, you can specify how to display missing data. Oracle Hyperion Financial Management interprets missing data as zero for display on reports and for calculating summary periods. You can specify whether a zero for missing data is interpreted as zero for the current period (Periodic) or as zero for year-to-date (YTD).

You also enable process management options by scenario. You can select whether to enable process management, select the maximum level of reviews for process units, and select the start year for phased submissions.

**Table 4-6 Scenario Member Attributes**

Attribute	Description
ConsolidateYTD	<p>Specifies the data view for consolidation - Year-to-Date or Periodic. This attribute is required. Specify Y for YTD or N for Periodic.</p> <p>If you set ConsolidateYTD to N, you must also set both ZeroViewForAdj and ZeroViewForNonadj options to Periodic.</p>
DefaultFreq	<p>Specifies the types of periods for which data input is valid for the scenario. This attribute is required.</p> <p>For example, a value of Monthly indicates that you can extract input data only in month-based periods, not in quarter-based or year-based periods. The frequency must be defined in the application profile.</p>
DefaultParent	<p>Specifies the default parent for the scenario.</p>
DefaultView	<p>Specifies the data view (Year-to-Date or Periodic) to use when Scenario View is selected in the point-of-view bar. This attribute is required. Specify YTD or Periodic.</p> <p>If you change the default view for a scenario and line-item detail has been entered, you should first extract the line-item detail and save it. Then delete the line-item detail from the scenario before changing the view. You must change the extracted line-item detail to match the new default view before reloading it.</p>
DefFreqForICTrans	<p>Specifies the default frequency for intercompany transaction data. This attribute must be a valid frequency and can contain a maximum of 80 characters. The default for this attribute is blank.</p>
Description	<p>Specifies the description for the scenario. The description can contain up to 80 characters, including spaces.</p>
EnableDataAudit	<p>Specifies whether changes to data for the scenario should be tracked in the data audit log. This attribute for an account or a scenario determines what can be audited. Specify one of these values:</p> <ul style="list-style-type: none"> <li>• Y to automatically audit all accounts. Even accounts that have EnableDataAudit set to False will be audited.</li> <li>• O to audit only those accounts that have EnableDataAudit set to True.</li> <li>• N to disable auditing for all accounts.</li> </ul>
MaximumReviewLevel	<p>Specifies the maximum level of reviews for process units for the scenario. Each process unit can have up to 10 levels of review. Specify a review level from 1 to 10. This attribute is required.</p>

Table 4-6 (Cont.) Scenario Member Attributes

Attribute	Description
Member	<p>Specifies the name for the scenario. This attribute is required. The name must be unique and can contain up to 80 characters, including spaces, but cannot start with a space.</p> <p>Do not use these characters in the scenario name:</p> <ul style="list-style-type: none"> <li>• Asterisk (*)</li> <li>• At sign (@)</li> <li>• Comma (,)</li> <li>• Curly brackets ({})</li> <li>• Double quotation marks""</li> <li>• Minus sign (-)</li> <li>• Number sign (#)</li> <li>• Period (.)</li> <li>• Plus sign (+)</li> <li>• Semicolon (;)</li> <li>• Slash mark (/)</li> </ul>
PhasedSubmissionStartYear	<p>In applications for which process management phased submissions is enabled, specifies the start year for phased submissions.</p>
SecurityClass	<p>Specifies the name of a valid security class that defines users who can access the data for the scenario. Security class names can contain up to 80 characters. For example, a user with None access rights to a scenario can open journal periods for the scenario.</p>
SupportsProcessManagement	<p>Specifies whether the scenario supports Process Management. Specify one of these values:</p> <ul style="list-style-type: none"> <li>• Y to enable the Process Management without email alerts</li> <li>• N to disable the Process Management option</li> <li>• A to enable Process Management and email alerts</li> </ul>
UserDefined1, UserDefined2, UserDefined3	<p>Stores custom information for the scenario. You can enter a maximum of 256 characters. The UserDefined1, UserDefined2, and UserDefined3 functions retrieve the text stored in this attribute.</p>



Table 4-6 (Cont.) Scenario Member Attributes

Attribute	Description
UsesLineItems	<p>Specifies whether accounts can use line-item detail in this scenario. Specify Y if the scenario can accept line items or N if the scenario cannot accept line items.</p> <p>If you change this attribute after line-item detail is entered, the stored line item detail may no longer be valid for the scenario. These behaviors occur:</p> <ul style="list-style-type: none"> <li>• If the scenario accepted line items and now it cannot, the line-item detail stored in the database is no longer valid. Only the total is displayed.</li> <li>• If the scenario did not accept line items and now it can, there is a total amount but no corresponding line-item detail information for the scenario. You can extract the total and then load it as line-item detail data so that the total matches the line-item detail information.</li> <li>• If the scenario does not accept line item detail initially, and if there is data present for the scenario, then it is not possible to enable line item details for the scenario or for any accounts in the scenario. To enable line items for that scenario, you must delete existing data, enable the line items in metadata and reload the data into the application.</li> </ul>
ZeroViewForAdj	<p>Specifies how to interpret missing, adjusted data values for the period. This attribute is required. Specify YTD or Periodic.</p> <p>If you set ConsolidateYTD to N, you must also set both ZeroViewForAdj and ZeroViewForNonadj options to Periodic.</p>
ZeroViewForNonadj	<p>Specifies how to interpret missing, nonadjusted data values for the period. This attribute is required. Specify YTD or Periodic.</p> <p>If you set ConsolidateYTD to N, you must also set both ZeroViewForAdj and ZeroViewForNonadj options to Periodic.</p>

## Defining Application Settings

Application settings apply to an entire Oracle Hyperion Financial Management application. Application settings determine the following information for the application:

- Is the organization dynamic, using organization by period?
- Which dimensions are secured?
- What default translation rates and methods are used?
- What is the ICP weight?
- Are consolidation rules applied?
- What is the default currency?

You define application settings by using the attributes in the Application Settings Attributes table.

**Table 4-7 Application Settings Attributes**

Attribute	Description
ConsolidationRules	Specifies whether consolidation rules are supported. Specify one of these values: Y to use the rules written in the Consolidate() routine in a user-defined rule. R to derive the proportional value in the Value dimension. Note that the proportional data is not stored. N to use the default consolidation and eliminations.
DefaultCurrency	Specifies the default currency for the application. This attribute is required.
DefaultRateForBalanceAccounts	The account that contains the translation rate to use for ASSET or LIABILITY accounts. This attribute is required.
DefaultRateForFlowAccounts	The account that contains the translation rate to use for REVENUE or EXPENSE accounts. This attribute is required.
DefaultValueForActive	Specifies the default value for the Active account. This attribute is required. Specify 0 if the child is considered inactive and is not consolidated into the parent. Specify 1 if the child is considered active and is consolidated into the parent.
EnableMetadataSecurityFiltering	Specifies whether users can view all dimension members or only dimension members to which they have access. The system filters these dimension members: <ul style="list-style-type: none"> <li>• Scenario</li> <li>• Entity</li> <li>• Intercompany Partner (ICP)</li> <li>• Account</li> <li>• Custom</li> </ul> Specify Y to filter out the dimension members to which the user does not have access. The default for this attribute is N.
FDMAppName	Name of the Oracle Hyperion Financial Data Quality Management application
ICPEntitiesAggregationWeight	Specifies the percentage of intercompany partner entity [ICP Entities] amounts that aggregate to the [ICP Top] member of the Value dimension. This attribute is required. The percentage is scaled to hundreds, with 1.0 equalling 100 percent.
MaxCellTextSize	Specifies the maximum number of characters that can be used for cell text. Specify 1900 or greater up to 2,147,483,646. Values between 0 and 1899 are not valid. The default value is 8,000.
MaxNumDocAttachments	Specifies the maximum number of document attachments per user. Specify -1 for no limit or a positive number up to 2,147,483,647. The default value is -1.

**Table 4-7 (Cont.) Application Settings Attributes**

Attribute	Description
MaxDocAttachmentSize	Specifies the maximum number of bytes for the size of document attachments. Specify -1 for no limit or a positive number up to 2,147,483,646. The default value is -1.
NodeSecurity	Specifies the type of security access for nodes. This attribute is required. Select Entity to check node data based on the entity's security access, or select Parent to check node data based on the parent's security access.
OrgByPeriodApplication	Specifies whether new consolidation structures can coexist with past consolidation structures in the application. Specify Y to allow new organizational structures or N to allow only current organizational structures.
SupportSubmissionPhaseforAccounts	Specifies whether phased submissions in process management are supported for accounts in the application. Valid values are Y or N. Default is N.
SupportSubmissionPhaseforCustom	Specifies whether phased submissions in process management are supported for the Custom members in the application. Valid values are Y or N. Default is N. The number of attributes is based on the number of Custom dimensions defined for the application. The attribute name changes to reflect the Custom dimension alias. For example, Custom in the attribute name is replaced with the Custom dimension alias.
SupportSubmissionPhaseforICP	Specifies whether phased submissions in process management are supported for ICP members in the application. Valid values are Y or N. Default is N.
UsePVAForBalanceAccounts	Specifies the default translation method for BALANCE accounts. Specify Y to use the periodic value (PVA) translation method or N to use the value at exchange rate (VAL) translation method.
UsePVAForFlowAccounts	Specifies the default translation method for FLOW accounts. Specify Y to use the periodic value (PVA) translation method or N to use the value at exchange rate (VAL) translation method.
UseSecurityForAccounts	Specifies whether accounts in the application are protected by security. Specify Y for security on accounts or N for no security.

**Table 4-7 (Cont.) Application Settings Attributes**

Attribute	Description
UseSecurityForCustom	Specifies whether Custom dimensions in the application are protected by security. Specify Y for security on Custom dimensions or N for no security on Custom dimensions.  The number of attributes is based on the number of Custom dimensions defined for the application. The attribute name changes to reflect the Custom dimension alias. For example, Custom in the attribute name is replaced with the Custom dimension alias.
UseSecurityForEntities	Specifies whether entities in the application are protected by security. Specify Y for security on entities or N for no security on entities.
UseSecurityForICP	Specifies whether ICP members in the application are protected by security. Specify Y for security on ICP members or N for no security on ICP members.
UseSecurityForScenarios	Specifies whether scenarios are protected by security. Specify Y for security on scenarios or N for no security on scenarios.
UseSubmissionPhase	Specifies whether phased submissions in process management are used in the application.  Valid values are Y or N. Default is N.
ValidationAccount	Specifies the account name to use for validation. The account used for validation must be an existing account.  In process management, validation accounts are used to ensure that the value equals zero before a process unit can be promoted to the next review level. Validation Account 1 is used for Submission Phase 1, and Validation Accounts 2 to 9 are used for Submission Phases 2 to 9.

## Organization by Period

Organizational structures can change for many reasons, including acquisitions, disposals, mergers, and reorganizations. The Organization by Period functionality enables the most recent consolidation structure to coexist with past structures in the same application.

To support organizational changes, Oracle Hyperion Financial Management uses the Active system account as a filter of the entity hierarchy. The Active account is an intercompany account that stores data at the parent level and uses the ICP dimension to store information about children. It specifies whether the consolidation status of a child entity into its parent is active or inactive.

For an ICP member that corresponds to a child of a parent, the Active account indicates to the system whether the child should be considered an active consolidation member for the current year, scenario, and period. Children that correspond to ICP members for which the Active account is equal to 0 are considered inactive children and are not consolidated. Children that correspond to ICP members for which the Active account is equal to 1 are considered active children and are consolidated. Changes to active child data affect the parent; changes to

inactive child data do not affect the parent. You can view or change Active account values in data grids.

The DefaultValueForActive attribute controls the status of children for which the Active account is blank. Therefore, you do not need to specify every parent-child intersection as active or inactive. By default, every child is active in relation to its parent unless otherwise specified.

## Defining Consolidation Methods

You define consolidation methods for an application by using the attributes in the Consolidation Methods Attribute table.

**Table 4-8 Consolidation Methods Attributes**

Attribute	Description
ConsolMethod	<p>Specifies the name for the consolidation method. This attribute is required. The name must be unique and can contain up to 80 characters, including spaces.</p> <p>Do not use these characters in the name:</p> <ul style="list-style-type: none"> <li>• Asterisk (*)</li> <li>• At sign (@)</li> <li>• Comma (,)</li> <li>• Curly brackets ({})</li> <li>• Double quotation marks ""</li> <li>• Minus sign (-)</li> <li>• Number sign (#)</li> <li>• Period (.)</li> <li>• Plus sign (+)</li> <li>• Semicolon (;)</li> <li>• Slash mark (/)</li> </ul>
Control	<p>Specifies the threshold that corresponds to the type of control to be used by the calculation routine. Specify one of these values for this attribute:</p> <ul style="list-style-type: none"> <li>• Blank</li> <li>• No</li> <li>• Limited</li> <li>• Full</li> </ul>
Description	<p>Specifies the description for the consolidation method. The description can contain up to 80 characters, including spaces.</p>
IsHoldingMethod	<p>Specifies whether the consolidation method is used for the holding company. This attribute is optional. Specify Y to use the method for the holding company, or N to use a different method for the holding company.</p>

**Table 4-8 (Cont.) Consolidation Methods Attributes**

Attribute	Description
PercentConsol	Specifies the consolidation percentage applied by the ownership calculation process. Specify a value for the percent (such as 100) or one of these keywords: <ul style="list-style-type: none"> <li>• POWN</li> <li>• POWNMIN</li> </ul> For information on POWN and POWNMIN, see <a href="#">Using POWN or POWNMIN Consolidation Methods</a> .
ToPercentControl	Specifies the upper boundary of the range for PercentControl. Used for the ownership calculation routine. Specify a value between 0 and 100. One of the method records must have a value of 100.
ToPercentControlComp	Specifies whether the upper boundary of the range of percent control is included in the range. Used for the ownership calculation routine together with the ToPercentControl attribute. This attribute is optional if the UsedByCalcRoutine attribute is N. Specify < or <= for this attribute.
UsedByCalcRoutine	Specifies whether the method is used for the automatic ownership calculation routine. Specify Y to use the method for ownership calculations, or N if you do not want to use the method for ownership calculations.

## Using Consolidation Methods

Consolidation methods are used during the consolidation and calculate ownership processes.

When you define consolidation methods in metadata, the system automatically generates the [ConsolMethod] system list for the From Currency dimension, which consists of all methods defined in the consolidation methods section.

There are two ways to assign the consolidation method to an entity for use during consolidation:

- You can manually assign the method through data load or data entry.
- You can automatically assign the method in the Calculate Ownership process, which is based on the ultimate percent control assigned to the entity. For details on calculating ownership, see the *Oracle Hyperion Financial Management User's Guide*.

## Assigning Consolidation Methods Manually

To enter consolidation method information manually, you can create a data grid with this information:

POV: Scenario, Year, Period, View, Entity, Value, Account, Custom

Scenario: Applicable scenario

Year: Applicable year

Period: Applicable period

Entity: A parent entity

Value: [None]

Account: [Method] system-generated account

Custom: ToCurrency

Row: ICP entities. For parent entities, you use the ICP Entities system list, or a user-defined list of selected ICP entities.

Column: FromCurrency. You should use the ConsolMethods system-generated list.

Method assignment information is stored in the account method of the data file of the parent entity. For each child of a parent, the system stores the consolidation method assignment in the ICP dimension. The assigned method is used when the children are consolidated to the parent.

For an intersection of the grid, use 1 to indicate the method assignment to the ICP entity. For example, if a parent group has two children, A and B, and you assign the Global method to A and the Equity method to B, enter 1 in the intersection for the Global method and entity A and 1 in the intersection for the Equity method and entity B.

## Using POWN or POWNMIN Consolidation Methods

The Calculate Ownership process uses settings in the consolidation method table to calculate the percentage of control and ultimate percentage of ownership, automatically assign the percentage of consolidation, and assign the methods for consolidation.

For the consolidation method corresponding to the EQUITY process, you use the POWNMIN keyword in the consolidation method table. When you use POWNMIN, the percentage of

consolidation that is assigned for the EQUITY company corresponds to the percentage used in a consolidation process that is performed in stages.

### **POWNMIN Calculation**

$$\text{POWNMIN} = \text{POWN} + \text{Sum of (Percent Minority of Entity Owners * Direct Percentage of Ownership in the Entity)}$$

Where:

- Percent Minority = Percent Consolidation – Percent Ownership
- Entity Owners are any entities within the descendants of the current parent that own shares of the entity being processed
- Direct Percentage of Ownership in the entity is retrieved from the Shares%Owned system account

Example:

- B is owned by A: 80%
- C is owned by A: 70%
- D is owned by B: 20%
- D is owned by C: 20%

The Parent entity GROUP has entities A, B, C, and D as dependents (A is the holding company). The system calculates percent ownership as follows:

- A: 100%
- B: 80%
- C: 70%
- D: 30%

Suppose the Percent Consolidation of D (from the consolidation method table) is POWNMIN.

If the consolidation process is done in stages, the POWNMIN process would be:

1. Consolidation of D into B using direct ownership percentage: 20%
2. Consolidation of D into C using direct ownership percentage: 20%
3. Consolidation of B and C into A using their respective percentages: (80% and 70%)

It is calculated as follows:

$$\text{Entity D's Percent consolidation} = 30\% + (100\% - 80\%) * 20\% + (100\% - 70\%) * 20\% = 40\%$$

Using this staged consolidation process, Entity D is consolidated using a total percentage of 40%.

When the subholdings B and C are consolidated into A, some minority interests corresponding to 10% are calculated on the Equity from Entity D.

### **POWN Calculation**

However, if the consolidation is done using a flat hierarchy, the process typically uses the ultimate percentage of ownership (POWN) as percentage of consolidation for the Equity



company. In this case, the percentage of consolidation for D into the Group would be 30%. No minority interests would be calculated on the Equity from Entity D.

In summary:

- Using POWN, the percentage of consolidation assigned to Entity D would be 30% (ultimate percentage of ownership).
- Using POWNMIN, the percentage of consolidation assigned to Entity D would be 40% (using a staged consolidation process).

## Defining Currencies

Currencies store translated values for entities. Every application must include a Currency dimension. The Currency dimension must include a currency for each default currency assigned to an entity in the Entity dimension. Each currency added to the Currency dimension is displayed as a system-generated member in the Value dimension. You can select a currency from the Value dimension to view data values translated to the currency.

You define currencies for an application by creating a dimension with the Currency dimension type, and by using the attributes in the Currency Attributes table. You create members in the Currency dimension for each currency needed in your application.

**Table 4-9 Currency Attributes**

Attribute	Description
Currency	<p>Specifies the name for the currency. This attribute is required. The name must be unique and can contain up to 80 characters, including spaces.</p> <p>Do not use these characters in the currency name:</p> <ul style="list-style-type: none"> <li>• Asterisk (*)</li> <li>• At sign (@)</li> <li>• Comma (,)</li> <li>• Curly brackets ({})</li> <li>• Double quotation marks ""</li> <li>• Minus sign (-)</li> <li>• Number sign (#)</li> <li>• Period (.)</li> <li>• Plus sign (+)</li> <li>• Semicolon (;)</li> <li>• Slash mark (/)</li> </ul>
Description	<p>Specifies the currency description. The description can contain up to 80 characters, including spaces.</p>
DisplayInICT	<p>Specifies whether currencies display in the drop-down list in the Intercompany Transactions module. Specify Y to display currencies or N to not display currencies. The default is Y.</p>

Table 4-9 (Cont.) Currency Attributes

Attribute	Description
Scale	<p>Specifies the unit in which amounts are displayed and stored for the currency by identifying where the decimal point is placed. This attribute is required.</p> <p>Also determines how the exchange rate must be entered. For example, if data is scaled to thousands, a value of 1 entered on a data form is stored as 1,000 in the database. Scale is a currency attribute, not an entity attribute. Specify one of these values for this attribute:</p> <ul style="list-style-type: none"> <li>• Blank = None</li> <li>• 0 = Units</li> <li>• 1 = Tens</li> <li>• 2 = Hundreds</li> <li>• 3 = Thousands</li> <li>• 4 = Ten thousands</li> <li>• 5 = Hundred thousands</li> <li>• 6 = Millions</li> <li>• 7 = Ten millions</li> <li>• 8 = Hundred millions</li> <li>• 9 = Billions</li> </ul>
TranslationOperator	<p>If you are using intercompany transactions, specifies whether to multiply or divide the local currency by the exchange rate. The default is blank. Specify one of these values:</p> <ul style="list-style-type: none"> <li>• D to calculate the local currency by dividing the transaction currency by the rate</li> <li>• M to calculate the local currency by multiplying the transaction currency by the rate</li> <li>• Blank to default the value to D</li> </ul>

The [Currencies] system member list is available for the From Currency and To Currency dimensions. Currencies that you add to the application are added to the [Currencies] member list. The [Currencies] list enables you to enter currency translation rates for pairs of currencies and filter out non-currency members.

## Defining Cell Text Labels

You can add cell text for any valid cell in a data grid or form. At times, you may need multiple cell text entries to store different types of information for a Point of View intersection. You can create multiple cell text entries and define cell text labels to easily identify types of information.

Before users can enter multiple cell text entries, the administrator must define cell text labels. The labels are loaded as metadata. They apply to all account cells in the application, and are available for selection when users enter cell text information. See the *Oracle Hyperion Financial Management User's Guide*.

These are the guidelines for defining cell text labels:

- The label name can be a maximum 80 characters. It must be unique within a data cell.
- It can contain spaces but cannot start with a space. If you are using an Oracle database, labels cannot contain spaces.

- The label name cannot contain these characters:
  - Ampersand (&)
  - Asterisk ( \* )
  - At symbol ( @ )
  - Comma ( , )
  - Curly brackets ( { } )
  - Double quotation mark ""
  - Forward slash (/)
  - Less than symbol (<)
  - Minus sign (-)
  - Number sign ( # )
  - Period (.)
  - Pipe character (|)
  - Plus sign (+)
  - Semicolon ( ; )
  - tilde character (~)

You can load cell text labels as part of a metadata load in a Cell Text Labels section in the load file. The following example shows a sample load file section for cell text labels:

```
<MISC Name="CellTextLabel">
```

```
<MISCENTRY>
```

```
<LABEL>MaturityDate</LABEL>
```

```
</MISCENTRY>
```

```
<MISCENTRY>
```

```
<LABEL>CouponRate</LABEL>
```

```
</MISCENTRY>
```

```
<MISCENTRY>
```

```
<LABEL>ExchangeRate</LABEL>
```

```
</MISCENTRY>
```

```
<MISCENTRY>
```

```
<LABEL>Rating</LABEL>
```

```
</MISCENTRY>
```

```
<MISCENTRY>
```

```
<LABEL>InterestRate</LABEL>
```

```
</MISCENTRY>
```

```
</MISC>
```

## System-Generated Accounts

When you create an application, system accounts for consolidation and ownership are automatically created for the application.



### Note:

You can change only the description, security class, and decimal location for system accounts. All other attributes for system accounts are predefined and cannot be modified.

## Consolidation Accounts

These Consolidation system accounts are required for each parent in the Entity dimension and are used in the consolidation process.



### Note:

All system accounts that are used for consolidation, except for the Active account, are BALANCE accounts. The Active account is a BALANCERECURRING account.

**Table 4-10 System Accounts for Consolidation**

Account	Description
Active	Consolidation status of a child into its parent. Yes if the child is consolidated into its parent; No if the child is not consolidated into its parent.
[PCON]	Percent consolidation. The percentage of the value of an entity that consolidates to the parent of the entity. Positive or negative numbers between -100 and 100, including 0. Default value is 100. For subsequent periods, derived as 0. Therefore, you must enter the percentage in all subsequent periods.
[POWN]	Percent ownership based on the shares of the entity that are owned by other entities. A positive number between 0 and 100. Default value is 100.
[DOWN]	Percent of direct ownership. A positive number between 0 and 100. Default value is 100.
[PCTRL]	Percent control based on the voting shares of the entity that are owned by other entities. A positive number between 0 and 100. Default value is 100.
Method	Consolidation method assigned to the entity. None or a selection from the list of available methods.
Consol1, Consol2, Consol3	Consolidation methods. A number between 0 and 255.

## Ownership Accounts

These Ownership system accounts are used for ownership calculations.



### Note:

All system accounts that are used for ownership calculations are BALANCE accounts.

**Table 4-11 System Accounts for Ownership**

Account	Description
SharesOwned	Total number of shares owned. Positive number or 0. Default is 0. Total shares owned must be less than or equal to the total shares outstanding.
VotingOwned	Number of voting shares owned. Positive number or 0. Default value is 0. Total voting shares owned must be less than or equal to the total voting shares outstanding.
SharesOutstanding	Total number of shares outstanding or the percentage of shares outstanding. Positive number or 0. Default value is 0. Enter the number of shares outstanding, or enter shares outstanding as a percentage. Enter 100 for percentage.
VotingOutstanding	Number of voting shares outstanding. A positive number or 0. Default value is 0. Enter the number of voting shares outstanding, or enter voting shares outstanding as a percentage. Enter 100 for percentage.
Shares%Owned	Calculated by system
Voting%Owned	Calculated by system

## Editing System-Generated Accounts

When you create an application, system account members are automatically created for the application.



### Note:

You can edit only the description, security class, and the decimal location for system account members. All other attributes are predefined and cannot be modified.

To edit system accounts:

1. Open the metadata file that contains the system-generated account members.

2. Select the **List View** tab.
3. From the list, select a system account, and modify the description, security class, or decimal location as needed.
4. Repeat step 3 as needed to modify other system account members.
5. Click **Save File**.

 **Note:**

You must load the updated metadata file into your application for your changes to take effect.

## Setting Up Intercompany Partners

Intercompany transactions are managed across the Intercompany Partner (ICP) dimension. The ICP dimension contains all intercompany balances that exist for an account. ICP is a reserved dimension used with the Account dimension and Custom dimensions to track and eliminate intercompany transaction details.

To set up an application for intercompany transactions, you must perform these actions:

- Indicate the accounts that perform intercompany transactions and indicate a plug account for each intercompany account (IsICP and PlugAcct attributes in account metadata)
- Indicate the entities that perform intercompany transactions (IsICP attribute in entity metadata)

When you create intercompany transactions, each group must have at least one intercompany account and one plug account. You designate an account as intercompany by selecting the IsICP attribute for the account. When an account is designated as intercompany and intercompany transactions are entered, eliminating or reversing entries are generated in the [Elimination] Value dimension member through the consolidation process.

A plug account is an account that, when eliminations are completed, stores the difference between two intercompany accounts in the Elimination Value dimension. A plug account can be set up as an ICP account. For a plug account to be detailed by ICP, set the IsICP metadata attribute to Y or R so the system writes eliminations to the corresponding ICP member. If you do not want a plug account to be detailed by ICP, set the IsICP attribute to N so the system writes eliminations to [ICP None].

During consolidation, transactions between valid intercompany entities are eliminated. See [Defining Entity Members](#).

The following table lists system-generated ICP elements.

**Table 4-12 System-Generated ICP Elements**

ICP Element	Description
[ICP Top]	Specifies the Top Intercompany member
[ICP None]	Specifies that no intercompany member is used
[ICP Entities]	Specifies the entities that are designated for intercompany transactions

## Editing System-Generated ICP Members

When you create an application, Intercompany Partner (ICP) members are automatically created for the application. An ICP member is created for each Entity member for which the IsICP attribute is selected.

 **Note:**

You can modify only the description and security class for ICP members. All other attributes are predefined and cannot be modified.

To modify intercompany members:

1. Open the metadata file that contains the system-generated Intercompany Partner members.
2. From the list, select an ICP member and, as needed, modify the description and security class.
3. Repeat step 2 as needed to modify other ICP members.
4. Click **Save File**.

 **Note:**

You must load the updated metadata file into your application for your changes to take effect.

## Editing System-Generated Value Members

When you create an application, Value members are automatically created for the application.

 **Note:**

You can modify only the description for Value members. All other attributes are predefined and cannot be modified.

After you load metadata, the system automatically creates three Value dimension members for each currency in your application:

- *CurrencyName*
- *CurrencyName Adjs*
- *CurrencyName Total*

Where *CurrencyName* is the currency label.

For example, for a currency of USD, the system creates these Value dimension members: USD, USD Adjs, and USD Total.



 **Note:**

The metadata file must have a description specified for the <Currency> Value member. If descriptions for currencies are not specified in the metadata file, when you load metadata, the currency descriptions are not displayed.

To modify a Value member description:

1. Open the metadata file that contains the system-generated Value members.
2. On the **Member Attributes** tab, in the hierarchy, select a member, and modify its description attribute.
3. Repeat step 2 as needed to add descriptions for other Value members.
4. Click **Save File**.

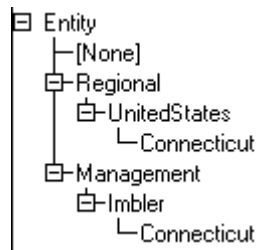
 **Note:**

You must load the updated metadata file into your application for your changes to take effect.

## Metadata Filtering Based on Security

When you filter metadata based on security, users see only the elements of the Scenario, Entity, ICP, Account, and Custom dimensions to which they have access. You set up metadata filtering at the application level by setting the AppSettings EnableMetadataSecurityFiltering metadata attribute to Y. For elements a user can view in a hierarchy, assign a security class and assign the user metadata access to the security class.

Users have implied access to the parents and ancestors of members to which they have access. With implied access, users see ancestors and parents in a hierarchical tree structure but cannot access them. For example, in the following tree structure, the user has access to only Connecticut even though the parents (UnitedStates and Imbler) and the ancestors (Management and Regional) are displayed in the tree.



## Creating Metadata Files of the APP Format

 **Note:**

To support characters beyond the standard ASCII character set in the app metadata file format:

- The application preference must be set to **"Save all files in Unicode format."**
- The app metadata file must be saved in UTF-16 LE byte-order mark (BOM) format before loading it into any application.

You can use an APP-format metadata file to add metadata to an application. The metadata file sections can be arranged in any order; however, the system automatically processes sections in this order:

- Currencies
- Entity dimension
- Scenario dimension
- Custom dimensions
- Account dimension
- Value dimension
- Intercompany Partner dimension
- Application settings
- Consolidation methods

For each type of dimension-related metadata, sections are processed in this order:

- Members
- Hierarchies

 **Note:**

For Custom dimensions, these sections must use the Custom dimension alias name.

Metadata for the Entity, Scenario, Account, and Custom dimensions is placed in the members and hierarchies sections. Custom dimensions may include the dimension section. The sections for each type of metadata can exist only once in a metadata file.

You can use these characters as a delimiter:

, ~ @ \$ % ^ | : ; ? \

 **Note:**

You must use a character that is not used in the file name or in any other way in the file. Delimiters are necessary only for ASCII files with the APP file extension. Delimiters are not necessary for extensible markup language (XML) files.

A line starting with an exclamation point (!) indicates the beginning of a new section in the metadata file and must be followed by a valid section name; for example, currencies, members, or hierarchies. True or false values are represented as Y for true or N for false. A line starting with an apostrophe (') is considered a comment line and is ignored by the system.

You can use these sections in a metadata file:

- File format
- Version
- Application settings
- Currencies
- Dimension
- Members
- Hierarchies
- Consolidation methods

## File Format

This section of a metadata file indicates the file version number. The version number changes only when changes are made to the file format. The file format is automatically generated when you extract metadata; if you are defining a file to load, you must include a valid file format. This syntax specifies the file format:

```
!FILE_FORMAT = majorNumber.minorNumber
```

*majorNumber* and *minorNumber* consist of one or two digits. *majorNumber* can contain a leading zero, and *minorNumber* can contain a trailing zero. You must include *majorNumber* and *minorNumber* and use only a period ( . ) as the decimal separator. These examples represent valid file format values:

```
!FILE_FORMAT = 11.12
```

```
!FILE_FORMAT = 11.120
```

```
!FILE_FORMAT = 011.120
```

```
!FILE_FORMAT = 011.12
```

## Version

This section of a metadata file indicates the version of Oracle Hyperion Financial Management that was used to extract metadata. The version number is automatically generated when you extract metadata; if you are creating a metadata file for loading, you do not need to specify a version. This syntax represents the version:

```
!VERSION = major version.minor version.build version
```

This example represents a valid version value:

```
!VERSION = 11.1.4749
```

## Application Settings

This section of a metadata file defines settings that apply to the entire Oracle Hyperion Financial Management application. For information on application settings attributes, see [Defining Application Settings](#).

This example specifies application settings attributes:

```
!APPLICATION_SETTINGS
```

```
DefaultCurrency=USD
```

```
DefaultRateForBalanceAccounts=Rate1
```

```
DefaultRateForFlowAccounts=Rate2
```

```
UsePVAForBalanceAccounts=Y
```

```
UsePVAForFlowAccounts=Y
```

```
ICPEntitiesAggregationWeight=1
```

```
DefaultValueForActive=1
```

```
ConsolidationRules=N
```

```
OrgByPeriodApplication=N
```

```
NodeSecurity=Entity
```

```
UseSecurityForAccounts=N
```

```
UseSecurityForEntities=Y
```

```
UseSecurityForScenarios=Y
```

```
UseSecurityForFlows=Y
```

```
UseSecurityForMarket=Y
```

```
UseSecurityForRegion=N
```

```
UseSecurityForCostCenter=N
```

```
UseSecurityForICP=N
```

## Currencies

This section of a metadata file defines currencies. This syntax specifies a currency:

```
Label; Scale; Descriptions
```

See [Defining Currencies](#).

This example specifies currency attributes:

```
!CURRENCIES

EURO;0;English=European Euro

GBR;0;English=Great Britain Pounds

USD;0;English=United Stated Dollars
```

## Members

This section of a metadata file defines the members of a dimension. You can use delimiters to represent missing values as empty. Enter dimension members by using this syntax:

```
!MEMBERS=Customers

'Label;IsCalculated;SwitchSignForFlow;SwitchTypeForFlow;
UserDefined1;UserDefined2;UserDefined3;SecurityClass;
DefaultParent;Descriptions

[None];N;N;N;;;;DefaultParent=#root

AllCustomers;Y;N;N;;;;DefaultParent=#root

Customer2;N;N;N;;;;DefaultParent=AllCustomers

Customer3;N;N;N;;;;DefaultParent=AllCustomers

Customer4;N;N;N;;;;DefaultParent=AllCustomers

Customer5;N;N;N;;;;DefaultParent=AllCustomers
```

**Note:**

For Custom dimensions, this section must use the Custom dimension alias name.

These topics list the formats for the members sections of the Account, Scenario, Entity, Custom, Value, and ICP dimensions.

## Account

Syntax for Account dimension members:

```
'Label, AccountType, IsCalculated, IsConsolidated, IsICP, PlugAcct,
CustomTop, NumDecimalPlaces, UsesLineItems, EnableCustomAggr, UserDefined1,
UserDefined2, UserDefined3, XBRLTags, SecurityClass, ICPTopMember,
EnableDataAudit, DefaultParent, Descriptions
```

See [Defining Accounts](#).

The Custom\_Order keyword is required for all new metadata files to define the columns for Custom dimensions. This section must match the Custom Order of the application profile (.per) file.

For example:

```
!Custom_Order=Product;Customers;Channel;UnitsFlows
```

The following example specifies attributes for two accounts, AdminExpenses and CapitalStock:

```
!MEMBERS=Account
```

```
AdminExpenses;EXPENSE;N;Y;Y; ;AllCustom3;AllCustom1;AllMarket;AllFlows;2;N;Y;Y
;Y;Y; ;
; ; ; ;N;DefaultParent=NetIncome
```

```
CapitalStock;LIABILITY;N;Y;N; AllCustom3;
[None];AllMarket;AllFlows;6;N;Y;Y;Y;Y;
; ; ; ; ;N;DefaultParent=TotalEquity;English=Capital Stock
```

## Scenario

Syntax for Scenario dimension members:

```
'Label, DefaultFreq, DefaultView, ZeroViewForNonadj, ZeroViewForAdj,
ConsolidateYTD, UserDefined1, UserDefined2, UserDefined3,
SupportsProcessManagement, SecurityClass, MaximumReviewLevel, UsesLineItems,
EnableDataAudit, EnableJournalsAutoLabel, DefFreqForPostingFlowTrans,
DefaultParent, Descriptions
```

See [Defining Scenario Members](#).

The following example specifies attributes for two scenarios, Actual and Budget:

```
!MEMBERS=Scenario
```

```
Actual;MTD;Periodic;Periodic;Periodic;N; ; ; ;N; ;10;Y;N;N;MTD ;DefaultParent=#root
```

```
Budget;MTD;Periodic;Periodic;Periodic;N; ; ; ;Y; ;10;Y;N;N;MTD ;DefaultParent=#root
```

## Entity

Syntax for Entity dimension members:

```
'Label, DefCurrency, AllowAdjs, IsICP, AllowAdjFromChildren, SecurityClass, UserDefined1, UserDefined2, UserDefined3, HoldingCompany, SecurityAsPartner, DefaultParent, Descriptions
```

See [Defining Entity Members](#).

The following example specifies attributes for three entities, California, Canada, and Connecticut:

```
!MEMBERS=Entity
```

```
California;USD;Y;Y;Y;US;;;;;DefaultParent=Imbler;English=State of California;French=California
```

```
Canada;USD;Y;N;N;;;;;DefaultParent=Regional
```

```
Connecticut;USD;Y;Y;N;US;Northeast;;;;;DefaultParent=Imbler
```

## Custom

Syntax for Custom dimension members:

```
'Label, IsCalculated, SwitchSignForFlow, SwitchTypeForFlow, UserDefined1, UserDefined2, UserDefined3, SecurityClass, DefaultParent, Descriptions
```

See [Defining Custom Members](#).





For example, suppose that you define the currencies USD and EUR with descriptions of "US Dollars" and "Euro", respectively. In addition, suppose you define these Value member descriptions in a loaded metadata file:

```
[None];English=ValueNone
```

```
<Currency Total>;English=Total
```

```
<Currency Adjs>;English=Adjs
```

```
<Currency>;English=Base
```

The following table describes the Value dimension member triplets that the system creates for the USD and Euro currencies.

**Table 4-13 Value Dimension Descriptions**

Value Member	Description
USD Total	US Dollars Total
USD Adjs	US Dollars Adjs
USD	US Dollars Base
EUR Total	Euro Total
EUR Adjs	Euro Adjs
EUR	Euro Base

 **Note:**

The metadata file must have a description specified for the Value member <Currency>. If descriptions for currencies are not specified in the metadata file, when you load metadata, the currency descriptions are not displayed.

## Intercompany Partner

You can use the members section to define security classes and descriptions for these system-defined members of the Intercompany Partner dimension:

- [ICP Top]
- [ICP None]
- [ICP Entities]

Syntax for Intercompany Partner members:

```
Label;SecurityClass;Descriptions
```

This example shows how to define descriptions for [ICP Top], [ICP None], and [ICP Entities] without specifying security classes:

```
[ICP Top];;English=Top ICP
```

```
[ICP None];;English=No ICP
```

```
[ICP Entities];;English=Entities ICP
```

## Consolidation Methods

This section of a metadata file defines the consolidation methods.

Syntax for consolidation methods:

```
Label;UsedByCalcRoutine;IsHoldingMethod;ToPercentControlComp;  
ToPercentControl;PercentConsol;Control;Descriptions
```

See [Defining Consolidation Methods](#).

This example specifies attributes for consolidation methods:

```
!CONSOLIDATION_METHODS
```

```
M2;Y;N;<=;20;0;No
```

```
M3;Y;N;<;50;POWN;Limited
```

```
M4;Y;N;<=;50;50;Limited
```

```
M1;Y;Y;<=;100;100;Full
```

```
M5;Y;N;<=;100;100;Full
```

## Hierarchies

This metadata file section defines parent-child relationships. A parent-child relationship is referred to as a node. A node can have its own set of attribute values.

A node record is a delimited list. The first two items of each line of the list identify a parent and child. You can use delimiters to represent missing attribute values as empty. All top-level members in a hierarchy should be represented as children of an empty parent.

 **Tip:**

Node records for Custom dimensions contain a third attribute. See [Custom Hierarchies](#).

To begin a hierarchies section, enter this line, replacing the <> characters with the dimension name:

```
!HIERARCHIES=<>
```

Do not include spaces when starting sections for Custom dimensions. For example, begin the Hierarchies section for the FLOW dimension with this line:

```
!HIERARCHIES=Customers
```

```
; [None];1
```

```
;AllCustomers;0
```

```
AllCustomers;Customer2;1
```

```
AllCustomers;Customer3;1
```

```
AllCustomers;Customer4;1
```

```
AllCustomers;Customer5;1
```

 **Note:**

For Custom dimensions, this section must use the Custom dimension alias name.

These topics list the formats for the Hierarchies sections of the Account, Scenario, Entity, and Custom dimensions.

## Account Hierarchies

Syntax for Account dimension hierarchies:

```
parentmemberlabel;childmemberlabel
```

This example specifies Account dimension hierarchies:

```
!HIERARCHIES=Account

; [None]

;ExchangeRates

ExchangeRates;Rate1

ExchangeRates;Rate2

;Plug

;NetProfit

NetProfit;NetIncome

NetIncome;GrossMargin

GrossMargin;Sales

GrossMargin;TotalCosts

TotalCosts;Purchases

TotalCosts;Salaries

TotalCosts;OtherCosts

NetIncome;AdminExpenses

NetIncome;InterestCharges

NetProfit;Taxes
```

## Scenario Hierarchies

Syntax for Scenario dimension hierarchies:

```
parentmemberlabel;childmemberlabel
```

This example specifies Scenario dimension hierarchies:

```
!HIERARCHIES=Scenario
```

```
;Actual
```

```
;Budget
```

## Entity Hierarchies

Syntax for Entity dimension hierarchies:

```
parentmemberlabel;childmemberlabel
```

This example specifies Entity dimension hierarchies:

```
!HIERARCHIES=Entity
```

```
:[None]
```

```
;Regional
```

```
Regional;UnitedStates
```

```
UnitedStates;California
```

```
California;Sunnyvale
```

```
California;FosterCity
```

## Custom Hierarchies

Syntax for Custom dimension hierarchies:

```
parentmemberlabel;childmemberlabel;AggregationWeight
```

This example specifies a Custom dimension hierarchy:

```
!HIERARCHIES=Products

; [None];1

;AllProducts;0

AllProducts;Golf;1

Golf;GolfBalls;1

Golf;GolfShoes;1

Golf;GolfTees;1

Golf;GolfClubs;1
```

## Dimensions Not Included in Metadata Files

The Year, Period, and View dimensions are not included in metadata files. You define these dimensions in the application profile that you specify when you define an application.

Value and Intercompany Partner dimension members are mostly system-defined. However, you can define descriptions for Value members, and security classes and descriptions for some members of the Intercompany Partner dimension.

- Value — Standard members are automatically generated. In addition, after you load metadata, the system automatically creates a triplet of Value dimension members for each currency that you loaded: *CurrencyName*, *CurrencyName Adjs*, and *CurrencyName Total*, where *CurrencyName* is the currency label. For example, for a currency of USD, Oracle Hyperion Financial Management creates these Value dimension members: USD, USD Adjs, and USD Total. You can define descriptions of the system-generated members, as well as descriptions appended to the Value members that the system creates for user-defined currencies.
- Intercompany Partner — This dimension is automatically generated. An Intercompany Partner dimension member is generated for each Entity dimension member in which the `IsICP` attribute is set to `TRUE`. You can define security classes and descriptions for some Intercompany Partner members as described in [Intercompany Partner](#).

## Metadata Referential Integrity

To prevent a referential integrity problem from occurring in the application, Oracle Hyperion Financial Management verifies that metadata changes are valid to the application in its current state before accepting the changes.

When you load metadata, the system compares the metadata load file with the metadata elements in the application. All changes are recorded, and some changes are checked against existing data. Modifications that cause referential integrity problems are not allowed.

 **Note:**

Locking data prevents stored data from being changed. It does not prevent aggregated data from changing if hierarchies are changed. To change a hierarchy total without affecting old aggregations, you should add a new hierarchy and retain the old hierarchy, rather than changing an existing hierarchy.

## Metadata Referential Integrity Checks

The log file provides information about specific changes in metadata attributes that require the system to check existing data with regard to the metadata file that you are loading.

The system also checks for invalid points of view between the load file and the metadata in the application. If a dimension member is not in the load file but exists in a journal in the application, the metadata load is prevented.

## Metadata Log File Referential Integrity Errors

In the metadata log file, referential integrity errors are displayed under this section:

```
Metadata referential integrity check started at
```

Each line in the referential integrity check section refers to a metadata integrity error in the load file. Errors found during the integrity check are displayed in this format:

```
Journals::SINGLECA1 Scenario::Actual Year::2014  
Value::[Contribution Adjs]  
Period::January has 1 occurrences of  
Changed::[SCENARIO::Actual::ZeroViewForAdj: Periodic to YTD]
```

This example shows that the metadata integrity error occurs in the SINGLECA1 journal with this point of view: Scenario Actual, Year 2014, Value [Contribution Adjs], Period January. The error is that the ZeroViewForAdj attribute for the Actual scenario was changed from Periodic to YTD. This change is not allowed because a journal exists for the Actual scenario.

## Using the Metadata Merge Utility

If you are upgrading to a new release from an existing release, you can use the Metadata Merge utility to merge your existing metadata files with the files in the latest version. This enables you to retain modifications that you made in your existing metadata file. The utility is located in the Financial Management\Utilities folder.



**Note:**

The Metadata Merge utility works only with metadata in XML format; it does not work for APP format.

**Usage**

```
MetadataMerge.bat -b<Base File>[i<Ignore descriptions>]-l <Latest file>-m
<Modified file>[-o <Output File>]
```

where:

**-b, -base <Base File>** = Base version of metadata file full path including the file name with extension

**-l, -latest <Latest File>** = Release version of metadata file full path including the file name with extension

**-m, -modified <Modified File>** = Modified metadata file full path including the file name with extension

**-o, -output <Output File>** = Output metadata file path, to which the updated metadata and Metadata Difference report will be saved

**-i, -ignoredesc <Ignore Descriptions>** = Ignore member description changes

**Example**

```
-b c:\temp\MetadataMerge\Comma_V1_B.xml
```

```
-l c:\temp\MetadataMerge\Comma_V2_R.xml
```

```
-m c:\temp\MetadataMerge\Comma_Customer_M.xml (User-modified metadata based on
Comma_V1_B.xml file)
```

```
-i true
```

```
Command: MetadataMerge.bat -b c:\temp\MetadataMerge\Comma_Metadata_B.xml -l
c:\temp\MetadataMerge\Comma_Metadata_R.xml -m
c:\temp\MetadataMerge\Comma_Metadata_M.xml -i true
```

To use the Metadata Merge utility:

1. Run `MetadataMerge.bat` from either File Explorer or the command line.
2. During the merge process, if there are any metadata conflicts, the system displays a warning message. Select one of these options:
  - Y - Yes. The system will apply changes from the latest release file and merge them into the existing file.
  - N - No. The system will not apply changes from the latest release file.
  - MA - Merge All. All changes will be applied from the latest release file. The system will not prompt you for further conflicts.
  - MN - Merge None. No changes will be applied. The system will not prompt you for further conflicts.

3. Copy the two images under the `Images` folder to the path where the Metadata Differences Report.html file is generated. These images are the Expand and Collapse icons for the tree in the Metadata Differences report.

## Loading Metadata

When you load a metadata file, Oracle Hyperion Financial Management replaces metadata with new metadata from the load file. Replacement is useful for making minor changes to the metadata, such as adding an account. For example, if your application includes a North America entity, and you load entities from a metadata file, the attributes for the North America entity in the file replace the attributes for the North America entity in the application.



### Note:

Do not use the ampersand character (&) in a metadata file. If you do, an error occurs.

When you load metadata files, the system waits for other tasks such as consolidation, data entry or other load processes to finish before proceeding to load the files. Oracle recommends that you load metadata during periods of light activity across the server cluster instead of, for example, during a long-running consolidation. You can check the Running Tasks page to see which consolidations or data loads, for example, are in progress.

Loading large metadata files can result in a proxy timeout error. If this error occurs, increase the Web proxy timeout setting.

After you load a metadata file to an application, users using the application are notified that the system has changed and that they must log off from the application and log back on.



### Caution:

You must delete orphaned members before loading; if orphaned members are not deleted, metadata is not updated.

You must select either the Merge or Replace load option. You can clear all metadata before loading the new metadata and also check the data integrity.

**Table 4-14 Metadata Load Options**

Load Option	Description
Merge	<p>If a dimension member exists in the load file and in the application database, then the member in the database is replaced with the member from the load file. If the database has other dimension members that are not referenced in the load file, the members in the database are unchanged.</p> <p>For example, a database contains entities CT, MA, and CA. You use the merge method to load a metadata file containing new information for CA only. In the database, CA is updated with the new information and MA and CT remain in the database and remain unchanged.</p>
Replace	<p>All dimension members in the application database are deleted and the members from the load file are put into the database.</p> <p>For example, a database contains entities CT, MA, and CA. You use the replace method to load a metadata file containing new information for CA only. In the database, CT and MA are deleted, and the only entity is CA with the new information from the load file.</p>
Clear All Metadata Before Loading	<p>All dimension members and corresponding data, journals, and intercompany transactions in the application database are deleted.</p> <p>If this option is selected, it overrides the function of the merge and replace methods.</p>
Check Integrity	<p>Checks the metadata against the data to ensure integrity. See <a href="#">Metadata Referential Integrity</a>.</p>

 **Note:**

Oracle recommends that you add Financial Management to the exceptions for your Web pop-up blocker. When you perform some tasks such as loading data, a status window pops up showing the task status. If you have a pop-up blocker enabled on your computer, the status window is not displayed.

To load metadata:

1. Open the application.
2. Select **Consolidation**, then **Load**, and then **Application Elements**.
3. In the **Metadata** section, enter the file name to load or click **Browse** to locate the file.

 **Note:**

By default, metadata files use the XML or APP file extension. The load process also accepts the TXT file extension, however, Oracle recommends that you use the XML or APP file extension.

**4. Optional: Select **Clear metadata and data**.** **Note:**

If you select this option, you cannot select elements in the Metadata Options section.

**5. Optional: Select **Check Integrity** to check the metadata file against the data in the current application.**

It is highly recommended that you select this option, because it ensures that the application is not adversely affected by the metadata in the load file.

 **Note:**

If integrity errors occur, they are written to the metadata log file, and no portion of the file is loaded into the application.

You must fix the errors before you can continue with this procedure. See [Metadata Log File Referential Integrity Errors](#).

**6. In the **Load Options** section, select a load method:**

- **Merge**
- **Replace**

**7. From **Delimiter**, select the character to be used to separate the metadata in the file.**

Delimiter characters are required only for ASCII files with the APP file extension. Delimiter characters are not required for XML files. These characters are valid:

, ~ @ \$ % ^ | : ; ? \

 **Note:**

Use a character that is not used in the file name or in any other way in the file. For example, if you use the comma in an entity description, you cannot use the comma as the delimiter.

**8. From the **Filters** section, select the types of metadata to load.**

 **Tip:**

To reset the filter selections, click **Reset**.

9. **Optional:** Click **Scan** to verify that the file format is correct.
10. Click **Load**.
11. **Optional:** To download the log file, click **Download Log**. Click **Open** to display the log file, or **Save** and select a location to save the file locally.

## Viewing Metadata Load Changes

After a metadata file has been successfully loaded, you can view detailed information about the metadata changes made during the load process. For example, if an attribute was changed in the Account dimension, the system displays the old attribute and the new attribute values. You can view a report of this information from the Task Audit module.

The metadata load changes report includes these details:

- Members added
- Members deleted
- Member attribute changes (does not include description changes)
- New parent/child relationships
- Deleted parent/child relationships
- Changes in aggregation weight for Parent-Child relationships in Custom dimensions

## Sample Metadata Load Report Output

\*\* Entity \*\*

Added member WestRegion.

Changed IsICP for China from Y to N.

Added Parent/Child Regional/Asia.

Removed Parent/Child Europe/Bulgaria.

\*\* Account \*\*


Renamed member SalesIC to SalesInterco.

Deleted member EastSales.

\*\* Scenario \*\*

Added member Forecast.

To view metadata load changes:

1. Select **Navigate**, then **Administer**, and then **Consolidation Administration**.
2. Select **Administration**, then **Audit**, and then **Tasks**.
3. From the Task Audit **Activity** column, click the Link icon  next to the Metadata Load process that you want to view.
4. Click **Open** and view the report with any text editor.

## Extracting Metadata

You can extract metadata to view or modify it. When you extract metadata, you save the file as an XML or APP file and specify the file name and location. After you modify metadata, you must reload the modified file into the application for the changes to take effect.

You cannot extract members of system-defined dimensions, such as the Value dimension. In addition, you cannot extract members of dimensions that are defined in application profiles, such as Year and Period.

To extract metadata:

1. Open the application.
2. Select **Consolidation**, then **Extract**, and then **Application Elements**.

3. In the **Metadata** section, from **Delimiter**, select the character to be used to separate the metadata in the file.

Delimiter characters are necessary only for ASCII files with the `.app` file extension. Delimiter characters are not necessary for XML files. These characters are valid:

, ~ @ \$ % ^ | : ; ? \

 **Note:**

Use a character that is not used in the file name or in any other way in the file. For example, if you use the comma in an entity description, do not use the comma as the delimiter.

4. Select a File Format:
  - **Financial Management (.app)**
  - **Financial Management (.xml)**
5. From **Filters**, select the types of metadata to extract.
6. Click **Extract**.
7. Follow the download instructions displayed in the browser to download the extracted file.

The instructions vary depending on the Web browser that you are using. Make sure to save the file in the Web directory that you set up.
8. **Optional:** To download the log file, click **Download Log**. Click **Open** to display the log file, or **Save** and select a location to save the file locally.

# 5

## Managing Member Lists

Member lists enable you to specify a subset of members within a dimension, and can reduce the time spent browsing the hierarchy of members. For example, if Italy, France, and UK are used frequently in the point of view for the Entity dimension, you can create a member list named European Region that includes the frequently used members. You can then select a European member from this list instead of browsing through the entire hierarchy of the Entity dimension.

You can also use member lists in rules. For example, you can write a rule that calculates all members of a member list.

Oracle Hyperion Financial Management uses two types of member lists:

- System-defined member lists
- User-defined member lists

System-defined member lists are generated after an application is created and metadata is loaded. These member lists group members according to common properties, such as all children of a specified parent, or all base members of a specified dimension. System-defined member list names are enclosed in brackets, for example [Descendants].

User-defined member lists contain members that you specify. You can create member lists for all Financial Management dimensions.

You can create static and dynamic member lists.

Static member lists contain user-specified dimension members. For example, you can create a static Account member list called ProfitAndLoss that includes only these accounts: Sales, Purchases, Salaries, OtherCosts, and TotalCosts. To add members to static lists, you must add the members manually.

Dynamic member lists contain members that, at runtime, meet specified criteria. Because the lists are built dynamically, they are updated when they are retrieved. For the Scenario, Year, Period, and Entity dimensions, you can use the current member in the POV as the starting point for the list.



### Note:

You can combine static and dynamic member lists in the same file.

Sample member list files are included when you install Sample Applications for Financial Management. The files are located in the Sample Applications folder in the directory to which you installed Financial Management.

## Creating Member List Files

You can use a text editor such as Notepad ++ to create member list files and then load the files into your application.



You can use Microsoft Visual Basic script syntax to add members to each static and dynamic list.

 **Note:**

The member list name cannot contain quotation marks (") or an ampersand (&).

By default, member list files use the LST file extension.

You use these subroutines to create member list files:

- Sub `EnumMemberLists ()`: Specifies which dimensions have member lists, and defines the member lists for each dimension. You define the number of lists for each dimension and the name of each member list within that dimension.
- Sub `EnumMembersInList ()`: Defines the members within each member list.

 **Note:**

The `HS.Dimension` function returns the dimension alias of the dimension. For Custom dimensions, in which you can specify a dimension name and dimension alias (long name), the `HS.Dimension` function returns the dimension alias (long name).

For information on adding dynamic member lists to the script, see [Dynamic Member Lists](#).

## EnumMemberLists

Each member list file must include an `EnumMemberLists` subroutine to specify which dimensions have member lists, the number of lists for each dimension, and the name of each member list. Within the `EnumMemberLists ()` subroutine, you use the syntax and functions in this table to define member lists:

Syntax	Description
<code>Dim <i>Element</i>Lists(<i>n</i>)</code>	Specifies the number of lists for each statement where <i>Element</i> is the dimension name for which you are creating member lists and <i>n</i> is the total number of member lists that you are defining for the dimension. For example, if the file contains three member lists for the Entity dimension, the syntax is as follows:  <code>Dim EntityLists(3)</code>
<code>HS.Dimension = "<i>Element</i>"</code>	where <i>Element</i> is the dimension name. For example:  <code>If HS.Dimension = "Entity" Then</code>

Syntax	Description
<code>ElementLists(n) = "ListName"</code>	<p>Specifies the name and numeric ID for each list where <i>Element</i> is the dimension name, <i>n</i> is the ID number of the member list, and <i>ListName</i> is the name of the list.</p> <p>You can use the @POV keyword to create a dynamic list based on the dimension member that is currently set in the POV. The entities appearing in the entity list can be based on the Scenario, Year, Period and Entity selected in the POV of a report.</p> <p>For example:</p> <pre>EntityLists(1) = "NewEngland"  EntityLists(2) = "Alloc"  EntityLists(3) = "AllEntities(@POV)"</pre>
<code>HS.SetMemberLists ElementLists</code>	<p>Sets the specified names and numeric IDs where <i>Element</i> is the dimension name. For example:</p> <pre>HS.SetMemberLists EntityLists</pre>

## EnumMembersInList

You use the EnumMembersInList subroutine to add members to a list. For a static member list, you list all members of the list in the script. Within the EnumMembersInList () subroutine, you use the syntax and functions in this table to define the members of each member list:

For all dimensions except Entity, you use the HS.AddMemberTo List statement, in which you must specify a member. For the Entity dimension, you use the HS.AddEntityToList statement, in which you must specify a member and its parent.

**Table 5-1 EnumMembersInList Syntax**

Syntax	Description
<code>HS.Dimension = "Element"</code>	<p>where <i>Element</i> is the dimension. For example:</p> <pre>If HS.Dimension = "Entity" Then</pre>

Table 5-1 (Cont.) EnumMembersInList Syntax

Syntax	Description
HS.MemberListID= <i>n</i>	Specifies the member list by its numeric ID where <i>n</i> = numeric ID assigned to the member list in the EnumMemberLists subroutine  HS.MemberListID = 1
HS.AddEntityToList <i>Member</i> HS.AddMemberToList <i>Member</i>	Adds members to a list for dimensions other than Entity where <i>Member</i> is the member name  HS.AddEntityToList "UnitedStates, "Maine"  HS.AddMemberToList "July"
HS.AddEntityToList <i>Parent, Member</i>	Adds members to a list for the Entity dimension where <i>Parent</i> is the parent of the member that you are adding, and <i>Member</i> is a member of the Entity dimension.  HS.AddEntityToList "UnitedStates", "California"
HS.Entity.List	Adds members to a list for the Entity dimension  HS.Entity.List (" "[Base]) "
HS.MemberListEntity HS.MemberListScenario HS.MemberListYear HS.MemberListPeriod	Use to specify a dynamic member list.

This section shows a sample of the EnumMembersInList section of the file. In this example, the entities for three entity lists are defined. The members of the Account list are also defined.

```
Sub EnumMembersInList()

If HS.Dimension = "Entity" Then

    If HS.MemberListID = 1 Then

        HS.AddEntityToList "United States", "Massachusetts"

        HS.AddEntityToList "United States", "Rhode Island"

        HS.AddEntityToList "United States", "Maine"

    ElseIf HS.MemberListID = 2 Then

        HS.AddEntityToList , "United States" "Connecticut"

    ElseIf HS.MemberListID = 3 Then

        HS.AddEntityToList "United States", "California"

    End If

ElseIf HS.Dimension = "Account" Then

    If HS.MemberListID = 1 Then

        HS.AddMemberToList "Sales"

        HS.AddMemberToList "Purchases"

        HS.AddMemberToList "Salaries"

        HS.AddMemberToList "OtherCosts"

        HS.AddMemberToList "TotalCosts"

        HS.AddMemberToList "GrossMargin"

        HS.AddMemberToList "HeadCount"
```



**Note:**

You can have as many member lists for each dimension as you need, and you do not need to create member lists for all dimensions.

## Dynamic Member Lists

For dynamic member lists, instead of listing all members of the member list, you enter rules to select members that meet specified criteria. Criteria are member properties such as currency or account type. The list is generated dynamically each time it is accessed by a user.

You can use Oracle Hyperion Financial Management functions and arguments to build member lists.

This syntax creates a dynamic member list to get all USD entities:

```
If HS.Dimension = "Entity" Then

    If HS.MemberListID=1 Then

        ELi=HS.Entity.List("", "")

    'Entities are read into an array.

        For i=Lbound(ELi) to Ubound(ELi)

    'Loops through all entities.

        If (StrComp(HS.Entity.DefCurrency(ELi(i)),
            "USD",vbTextCompare)=0) Then

            HS.AddEntityToList "",ELi(i)

    'String compares default currency for entity to USD. If there is a match, the
    entity is added to the member list.

            End If

        Next

    End If

End If
```

In this example, the lists of entities is received into an array. For each entity in the array, the value of the DefaultCurrency property is compared with the preferred value of USD. If the value is equal to USD, the entity is added to the list. The system then processes the next entity in the array.

## Dynamic POV Member List

Dynamic POV Member lists are created dynamically based on the current POV member of one or more dimensions.

The bold sections of this sample member list file show the dynamic POV sections.

```
Sub EnumMemberLists()  
  
Dim EntityLists(5)  
  
If HS.Dimension = "Entity"Then  
  
    EntityLists(1) = "AllEntities"  
  
    EntityLists(2) = "AppCur"  
  
    EntityLists(3) = "NoappCur"  
  
    EntityLists(4) = "Global(@POV) "  
  
    EntityLists(5) = "POWN(@POV) "  
  
    HS.SetMemberLists EntityLists  
  
End If  
  
End Sub  
  
Sub EnumMembersInList()  
  
If HS.Dimension = "Entity" Then  
  
    If HS.MemberListID = 1 Then  
  
        HS.AddEntityToList "", "Corp_Ops"  
  
        HS.AddEntityToList "", "China"  
  
        HS.AddEntityToList , "", "Germany"  
  
        HS.AddEntityToList "", "UK"  
  
    End If  
  
End If
```

```
    "..Y#" & YearPOV & ".P#" & PerPOV)

    If IsArray(EntList) Then

        For each Ent in EntList

            If Ent <> "[None]" Then

                If HS.Node.Method("S#" & ScenPOV & ".Y#" & YearPOV & ".P#" &
                PerPOV & ".E#" & EntPOV & "." & Ent)="GLOBAL" Then

                    If HS.MemberListID = 4 Then HS.AddEntityToList "",Ent

                End If

                If HS.Node.POwn("S#" & ScenPOV & ".Y#" & YearPOV & ".P#" & PerPOV
                & ".E#" & EntPOV & "." & Ent) > 0.5 Then

                    If HS.MemberListID = 5 Then HS.AddEntityToList "",Ent

                End If

            End If

        End If

    Next

End If

End If

End Sub
```

## Loading Member Lists

After creating a member list script file, you load it into your application. Before loading the file, you can scan to verify that it is formatted correctly. When you modify the file, you must reload it into the application.



When you load member list files, the system waits for other tasks such as consolidation, data entry or other load processes to finish before proceeding to load the files. Oracle recommends that you load member lists during periods of light activity across the server cluster instead of, for example, during a long-running consolidation. You can check the Running Tasks page to see which consolidations or data loads, for example, are in progress.

After you load a member list file into an application, users using that application are notified that the system has changed and that they must log off from the application and log back on. The calculation status also changes to OK SC (system changed).

 **Note:**

Oracle recommends that you add Oracle Hyperion Financial Management to the exceptions for your Web pop-up blocker. When you perform some tasks such as loading data, a status window pops up showing the task status. If you have a pop-up blocker enabled on your computer, the status window is not displayed.

To load member lists:

1. Open the application.
2. Select **Consolidation**, then **Load**, and then **Application Elements**.
3. In the **Member Lists** section, enter the file name to load, or click **Browse** to locate the file.

 **Note:**

By default, member list files use the LST file extension. The load process also accepts the TXT file extension, however, Oracle recommends that you use the LST file extension.

4. **Optional:** Click **Scan** to verify that the file format is correct.
5. Click **Load**.

 **Note:**

If an invalid member list is loaded, users may not be able to access the data grid. If the following error message is displayed, you may need to correct the member list and load the file:

```
EPMHFM-102: Error executing VBScript,  
An unexpected error has occurred. Check server log or contact your system  
administrator for more information.
```

6. **Optional:** To download the log file, click **Download Log**. Click **Open** to display the log file, or **Save** and select a location to save the file locally.

## Extracting Member Lists

You can extract member lists from an application. Extracted member lists are saved as ASCII files. By default, member list files use the LST file extension. After you extract member lists, you can view and modify the member list information in a text editor.

To extract member lists:

1. Open the application.
2. Select **Consolidation**, then **Extract**, and then **Application Elements**.
3. In the **Member Lists** section, click **Extract**.
4. Follow the download instructions displayed in the browser to download the extracted file.

The instructions vary depending on the Web browser that you are using. Make sure to save the file in the Web directory that you set up.

5. **Optional:** To download the log file, click **Download Log**. Click **Open** to display the log file, or **Save** and select a location to save the file locally.

## System Lists by Dimension

The following table lists the name of the system-generated list and the dimensions in which it can be used.

**Table 5-2 System Lists by Dimension**

System List	Scenario	Entity	Account	ICP	Custom	Value	Year	Period	
[Hierarchy]		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
[Descendants]		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
[Children]			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>
[Base]			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
[Parents]			<input checked="" type="checkbox"/>						
[Ancestors]			<input checked="" type="checkbox"/>						
[System]				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>				

Table 5-2 (Cont.) System Lists by Dimension

System List	Scenario	Entity	Account	ICP	Custom	Value	Year	Period
[Currencies]						C 1 C 2		
[Consolidation Method]						C 1		
[Inputs]							<input checked="" type="checkbox"/>	
[Adjustments]							<input checked="" type="checkbox"/>	
[Totals]							<input checked="" type="checkbox"/>	
[Default Currencies]							<input checked="" type="checkbox"/>	
[First Generation]								<input checked="" type="checkbox"/>
[Second Generation]								<input checked="" type="checkbox"/>
[Third Generation]								<input checked="" type="checkbox"/>
[Fourth Generation]								<input checked="" type="checkbox"/>
[Fifth Generation]								<input checked="" type="checkbox"/>
[Sixth Generation]								<input checked="" type="checkbox"/>

# 6

## Managing Journals

Many external general ledger systems can generate ASCII text files containing journal information that you can load into a Oracle Hyperion Financial Management application. If necessary, you can edit the ASCII file using a text editor before loading it.

Sample journal files are included when you install Sample Applications for Financial Management. The files are located in the Sample Applications folder in the directory to which you installed Financial Management.

### Creating Journal Files

You can create journal files using an ASCII format supporting multibyte character sets (MBCS) or a file encoded with the Unicode format, using Little Endian byte ordering. By default, journal files use the JLF file extension.

A security information file can contain these sections:

- File Format
- Version
- Journal Group
- Standard
- Recurring
- Header - Scenario, Year, Period

A line starting with an exclamation point (!) indicates the beginning of a new section in the journal file, and must be followed by a valid section name (for example, Year). A line starting with an apostrophe (') is considered a comment line and is ignored by the system.

You can use these special characters to separate information within the file as long as the character is not used in the file in another way:

Character	Description
&	ampersand
@	at sign
\	backslash
^	carat
:	colon
,	comma
\$	dollar sign
#	number sign
%	percent sign
	pipe sign
?	question mark

Character	Description
;	semicolon
~	tilde

 **Note:**

You must use the same delimiter character throughout the file. Using different delimiter characters within the same file causes an error when you load the file.

## File Format Section

This file section contains the file version number. This number only changes when changes are made to the file format. The file format is automatically generated when you extract journals.

 **Note:**

This section is not required.

This example specifies the file format:

```
!File_Format = 1.0
```

## Version Section

This file section contains the Oracle Hyperion Financial Management version that you used to extract journals. The version number is automatically generated when you extract journals.

 **Note:**

This section is not required.

This example specifies the version:

```
!Version = 11.1
```

## Journal Group Section

This file section uses this syntax to define journal groups.

```
!GROUP=<journal group>;<journal group description>
```

For example, this example defines two journal groups:

```
!GROUP=Allocations;Allocations Journals Group
!GROUP=Tax;Tax Journals Group
```

## Standard Section

Standard templates apply to all scenarios, years, and periods in an application. They are not dependent on a specific combination of scenario, period, and year.

This syntax specifies a standard template:

```
!STANDARD = <label>, <balancing attribute>, <type>, <journal group>,
<securityclass>, <SingleParent.SingleEntity>
```

```
!DESC=<journal description>
```

```
<parent.entity>, <account>, <ICP>, <CustomDimensionName>, <amount type>,
<amount>, <line item desc>
```

## Recurring Section

Recurring templates apply to all scenarios, years, and periods in an application. They are not dependent on a specific combination of scenario, period, and year, but are dependent on Value Adjs.



### Note:

You cannot create a recurring template for an auto reversing template. For the type attribute, the value must be **R** for regular.

This syntax specifies a recurring template:

```
!RECURRING = <label>, <balancing attribute>, <type>, <value>, <journal
group>, <securityclass>, <SingleParent.SingleEntity>
```

```
!DESC=<journal description>
```

```
<parent.entity>, <account>, <ICP>, <CustomDimensionName>, <amount type>,
<amount>, <line item desc>
```

## Header Section

This file section contains the scenario, year, and period information. The journal type header information and corresponding detail lines follow the Header section. This organizes journal

information according to a specific scenario, year, and period. You can specify multiple Header sections in the journal file.

This syntax specifies the scenario, year, and period:

```
!SCENARIO= Actual
```

```
!YEAR = 2014
```

```
!PERIOD = January
```

The Header section is followed by journal detail information for Actual, January, 2014.

The Journal subsection of the Header section provides journal detail information for the specified Scenario, Period, and Year.

[Table 1](#) describes the attributes that are used in the Journal subsection. These attributes are used for regular journals and recurring and standard templates.

**Table 6-1 Journal Attribute Descriptions**

Attribute	Value
<label>	User-defined label for journal, 20 characters maximum
<balancing attribute>	<ul style="list-style-type: none"> <li>• U = unbalanced</li> <li>• B = balanced in total</li> <li>• E = balanced by entity</li> </ul>
<type>	<ul style="list-style-type: none"> <li>• R = regular journals</li> <li>• A = auto-reversing journals</li> <li>• V = auto-reversal journal</li> </ul> <p>You cannot load system-generated auto reversals, but you can extract them.</p>
<status>	<ul style="list-style-type: none"> <li>• W = Working</li> <li>• S = Submitted</li> <li>• A = Approved</li> <li>• P = Posted</li> <li>• R = Rejected</li> </ul>
<value dimension>	<ul style="list-style-type: none"> <li>• [Contribution Adjs]</li> <li>• [Parent Adjs]</li> <li>• &lt;Entity Curr Adjs&gt;</li> <li>• &lt;Parent Curr Adjs&gt;</li> </ul>
<journal group>	Optional: User-defined parameter, 30 characters maximum Groups must be preloaded.
<security class>	Optional: Valid security class that is associated with the journal  If you do not assign a security class, the journal assumes the Default security class. Only users who have access rights to this security class can access the journal.

**Table 6-1 (Cont.) Journal Attribute Descriptions**

Attribute	Value
<singleparent.singleentity>	Optional: Valid parent/entity pair that is used by all the line items in the journal. When you specify a single parent/entity pair for the entire journal, the parent.entity attribute is not used.
<journal description>	Journal description, which can contain up to 255 characters. You can load descriptions with multiple lines, provided each line starts with this syntax: !DESC=
<parent.entity>	Valid member of the Entity dimension. The parent is required only for the Contribution Adjs, Parent Adjs, and ParentCurrency Adjs members of the Value dimension. This attribute is used only if the Single Parent.Single Entity attribute is not used.
<account>	Valid member of the Account dimension. For regular journals, this must be an input account and the account type must be REVENUE, EXPENSE, ASSET, LIABILITY, FLOW, or BALANCE.
<ICP>	Optional: Valid member of the Intercompany Partner dimension. This attribute is optional; however, you must at least specify ICP None.
<custom>	Optional: Valid members of the Custom dimensions
<amount type>	<ul style="list-style-type: none"> <li>• D = debit</li> <li>• C = credit</li> </ul>
<amount>	Positive amount regardless if the amount type is debit or credit
<line item description>	Optional: Description of the specific journal detail, which can contain up to 50 characters

This syntax specifies a regular journal:

```
!JOURNAL = <label>, <balancing attribute>, <type>, <status>, <value dimension>, <journal group>, <SecurityClass>, <SingleParent.SingleEntity>
```

```
!DESC=<journal description>
```

```
<parent.entity>, <account>, <ICP>, <CustomDimensionName>, <amount type>, <amount>, <line item desc>
```

## Loading Journals

You can load working, rejected, submitted, approved, and posted journals as well as standard and recurring journal templates. You cannot load automated consolidation journals because they are created by the consolidation process.

Before you can load journals, you must first open the periods to which to load journals. See "Managing Periods" in the *Oracle Hyperion Financial Management User's Guide*.



Journals are loaded using the Replace mode, which clears all data for a particular journal label before loading the new journal data. You must load posted journals to periods that are open. If you load a posted auto-reversing journal, an approved reversal is automatically generated in the next period, and you must manually post the generated reversal.

By default, journal files use the JLF file extension. The load process accepts other file extensions such as TXT or CSV, however, Oracle recommends that you use the JLF file extension.

When you change the default load options, the options are updated for all the rows. You can use the Override option to enable a specific row and update the options for that row.

When the load process is complete, a link displays for the log so that you can view any errors.

 **Note:**

Oracle recommends that you add Oracle Hyperion Financial Management to the exceptions for your Web pop-up blocker. When you perform some tasks such as loading data, a status window pops up showing the task status. If you have a pop-up blocker enabled on your computer, the status window is not displayed.

To load journals:

1. Select **Consolidation**, then **Load**, and then **Journals**.
2. For **Delimiter**, enter the delimiter used to separate the journals data in the file. These characters are valid:

, ~ @ \$ % & ^ | : ; ? \

 **Note:**

Use a character that is not used in the file name or in any other way in the file. For example, if you use the comma in an entity description, you cannot use the comma as the delimiter.

3. **Optional:** If you are using Firefox as the browser, you can load multiple files. Select **Multiple selection** if it is not already selected, and click **Browse** to locate the files.

You can select a maximum of 10 files at one time. The system then populates the detail rows with the selected files.

You can also drag and drop files into the File Selection Content area.

4. **Optional:** Click **Add** to add more rows for loading journals.

 **Tip:**

To delete a row, select a row and click **Delete**.

5. **Optional:** To override the default file settings for a row, click **Override**.

 **Tip:**

To reset the load options to the default values, click **Reset**.

- Click **Load**.

 **Note:**

If you reload existing files, the system displays a warning prompt asking if you want to use them again. If you do, click **Yes**.

## Extracting Journals

You can extract journals, recurring journals, and journal templates from an application. You can select the Scenario, Year, Period, Entity, and Value dimensions for which to extract journals. You can select the journal status, journal type, and journal balance type to extract.

When you extract journals, they are saved to an ASCII file that supports multibyte character sets (MBCS). By default, journal files use the JLF file extension. After you extract journals, you can view and modify the journal information in a text editor.

You can extract automated consolidation journals to external systems, however you cannot reimport them into Oracle Hyperion Financial Management. When you extract these journals, the Balance type is blank for automated consolidation journals.

When the extract process is complete, a link displays for the log so that you can view any errors.

To extract journals:

- Select **Consolidation**, then **Extract**, and then **Journals**.
- Specify a Scenario and Year from which to extract journals.
- Optional:** Select a Period from which to extract journals.
- Optional:** Select one or more Entity and Value dimensions to extract.

To extract automated consolidation journals, you must select [Proportion] and/or [Elimination] as the Value member.

 **Note:**

For the Period, Entity, and Value dimensions, if you do not select specific members, the system assumes that you want to extract all members for the dimension. However, if you specifically select members, the system displays a plus sign (+) next to the dimension to indicate multiple selections.

- For **Delimiter**, enter the delimiter used to separate the journals data in the file. These characters are valid:

, ~ @ \$ % & ^ | : ; ? \

 **Note:**

Use a character that is not used in the file name or in any other way in the file. For example, if you use the comma in an entity description, you cannot use the comma as the delimiter.

6. Select **Journal Types** to extract:

- **Templates**
- **Recurring Templates**
- **Journals**

7. **Optional:** Enter the Label and Group for the journals to extract.

You can use the percent sign (%) as a wildcard.

For automated consolidation journals, in the Label field, you can query on the Nature value of the audit transaction that you specified in the consolidation rule. You cannot use a number for the label.

8. Select the journal Status, Type, and Balance Type to extract, or select **All**.

9. Click **Extract**.

10. Follow the download instructions displayed in the browser.

The instructions vary depending on the Web browser that you are using. Make sure to save the file in the Web directory that you set up.

# 7

## Managing Data Forms

Data forms are templates with predefined rows and columns that you set up for users to view and enter specific data. Users cannot add rows and columns to or remove rows and columns from data forms. They can change point of view selections only for the dimensions that you specify when you set up the form. You can define links from one form to another to enable users to drill through to view data in the linked form.

To create data forms, you must be assigned to the Administrator or Manage Data Entry Forms security role. To load, extract, and delete data forms, you must be assigned to the Manage Data Entry Forms role.

You can create data forms in two ways:

- Using the Data Form options in the Form Designer
- Writing a script in the Script view

By default, data forms use the WDF file extension.

You can easily switch between the Designer and Script view. If the script contains any errors, they are displayed in the Script view. From the Script view, you can also print data forms.

You can switch from the Designer view of a data form to Open Form. The system prompts you to save the form, then displays the data form.

## Creating Data Forms in the Form Designer

You use the Form Designer to set the Point of View, and specify form details, row, column, and header options.

After you create a data form, you can scan it to check the validity. The system generates the form script and validates it. If there are errors, they are displayed in the Script view. See [Using Data Form Scripts](#).

For information on setting form details:

- [Setting the Point of View](#)
- [Specifying Form Details](#)
- [Specifying Form Row and Column Options](#)
- [Specifying Form Headers](#)

To create a data form:

1. Select **Consolidation**, and then **Documents**.
2. Click **New**, and then **Data Form**, or select **Actions**, then **New**, and then **Data Form**.

The Forms Designer is displayed by default.

 **Tip:**

If the Designer page is not automatically displayed, click **Designer**, or select **Actions**, and then **Designer**.

3. Set the Point of View for the data form.
4. Specify the Form Details.
5. Specify the Row and Column options.
6. **Optional:** To add rows or columns, click **Add New Row**, **Add New Column**, or select **Actions**, and then **Add New Row** or **Add New Column**.

 **Tip:**

To delete a row or column, click **Delete Column/Row**, or select **Actions**, and then **Delete Column/Row**.

7. Specify the Header options.

 **Tip:**

To reset the form to the default values or the last saved definition, click **Reset**.

8. **Optional:** Click **Scan** to verify the validity of the form.
9. Click **Save** to save the form, enter the information for the form, and then click **Save**.

## Setting the Point of View

You can define the Background POV and selectable POV using the POV bar. A Background POV for the form specifies for each dimension the initial value that is displayed on the data form. A Selectable POV for the form consists of the dimensions for which users can select members.

For any dimension in the POV bar, you specify the Background POV members by selecting a member. If you select a member list, it is used as a selectable POV, from which data form users can select members. You cannot select more than two items and more than one member or member list.

If you have defined dynamic POV member lists, you can use them to quickly select valid members for the selected dimension. In the Member Selection dialog box, the valid dynamic POV member lists for the selected dimension are displayed with an @POV suffix at the bottom of the dialog box.

These rules apply to using the POV:

- If you do not select a member or member list and the dimension is not used in a row or column, the system uses the member from the user POV for the initial value for the dimension.
- If you specify values in a Background and Selectable POV for a dimension, and the member for the Background POV is not in the member list for the Selectable POV, the

system uses the first member in the member list of the Selectable POV as the Background POV member.

- Users need full access to the member hierarchy to be able to work with all the members in the hierarchy. For example, if you want users to access all Custom 4 members, you must enable access to the parent entity, in this case, CustomTop.

The Form Designer grid contains rows and columns. In a new form, by default the Scenario dimension is used for the grid column, and the Period dimension is used for the grid row. You can add a column or row to the form, and then drag and drop dimensions from the POV bar to the grid rows and columns to define the data form layout. If you drag and drop more than one dimension in the same row or column, you can reorder the dimensions. When you click a dimension, you can select members from the Member Selection page to include in the grid or POV.

When you select a row or column in the grid, the corresponding row and column options are displayed in the Form Details property pane. You can include calc expression in a row or column. In this case, you cannot drag and drop dimensions in the row or column, instead you must enter the calc expression in the row or column using the property pane. Dimensions that are used for rows are not available for columns and vice-versa.

## Specifying Form Details

The Form Details section specifies form properties such as grid, print, display, and suppression options for the data form. You can include Instructions for users of the data form, and specify any on-demand rules that are available for the data form.

 **Note:**

The items in the Form Details section are only updated in the script if you change the default settings. If you leave the default settings unchanged, the keywords for these items do not display in the script.

For valid values for each option, see [Using Data Form Scripts](#).

## Specifying On-Demand Rules for Data Forms

You can create on-demand rules that can be run from data forms. On-demand rules are useful when you want to run only a subset of calculations to quickly see the results in the data form. For example, when you are working in a data form that has been set up to run a specific on-demand rule, you can enter data and then run the on-demand rule to quickly see the results of the calculation.

You create on-demand rules in the application rule file. The rules are created in a new subroutine and are identified by the OnDemand prefix, for example, OnDemand\_Calculation. See [Creating Rules Files](#).

To specify on-demand rules for a data form:

1. Select **Consolidation**, and then **Documents**.
2. Click **New**, and then **Data Form**, or select **Actions**, then **New**, and then **Data Form**.

The Forms Designer is displayed by default.

 **Tip:**

If the Designer page is not automatically displayed, click **Designer**, or select **Actions**, and then **Designer**.

3. In the **Form Details** panel, for **On Demand Rules**, use one of these methods:
  - If you know the rule names, enter them in the text box in a comma-separated list.
  - To search for available rules, click the **Edit** button next to the text box, then from the **Available Rules** pop-up dialog box, select one or more rules to use in the form and click **OK**.
4. Click **OK**.
5. Save the data form.

## Specifying Form Row and Column Options

The dimension elements that you specify for rows and columns override elements that are set in the Background POV or Selectable POV. You can specify only one member list per row or column.

For valid values for each option, see [Using Data Form Scripts](#).

If a form requires more than 100 rows, you can use the Scripts tab to specify the additional rows. For information on the syntax to use, see [Rn](#).

If a form requires more than 24 columns, you can use the Scripts tab to specify the additional columns. For information on the syntax to use, see [Cn](#).

In addition, you can specify Calculated Row or Column, and Calc Expressions for rows and columns. When you select the Calculated Row option, the selected row or column becomes a calculated row and column, and you must enter a calc expression.

 **Note:**

You can use Other to specify syntax for row or column keywords not displayed in the Options section, such as Blank and ReadOnly. In addition, if the definition of a row or column in the script contains deprecated or invalid syntax, that syntax is displayed in Other.

## Specifying Form Headers

The Headers section enables you to specify header properties for each dimension, such as Show Label, Description, or Both, specify a Style, specify the maximum length of the labels or descriptions for members and member lists, and whether the lengths are fixed. You can specify different lengths for different dimensions.

The Headers section also contains the Other text box, which has two uses:

- If syntax is added for the [HeaderOption](#) keyword, you can specify that syntax in Other.
- If the form contains invalid syntax for the `HeaderOption` keyword for a dimension, the Other text box displays that syntax.

For valid values for each option, see [Using Data Form Scripts](#).

## Using Data Form Scripts

The Script feature enables you to view, modify, and print the script. You enter the syntax in the text box on the Script page.

You can use three types of syntax elements when creating data form scripts: keywords, values, and options. Keywords are on individual lines in the script and are placed to the left of the equal sign. Values are placed immediately after the equal sign to complete the line. Options can be added to a line of script where each is delimited by a comma.

 **Note:**

When a value is required, it must be specified before options. Options are never required and can be in any order.

Sample data form scripts are installed with Sample Applications for Oracle Hyperion Financial Management. The files are located in the Sample Applications folder in the directory to which you installed Financial Management.

 **Note:**

Data entry form script elements are not case-sensitive.

**Table 7-1 Data Form Script Syntax**

Script Syntax	Description
<a href="#">AddMember</a>	Use in a row definition to allow the user to add data for a member that previously had no data or contained zeros and was suppressed from the form. The option adds an icon to the form that, when clicked, enables the user to select members to add to the form.
<a href="#">BackgroundPOV</a>	Use to specify the background dimension members for the form.
<a href="#">Blank</a>	Use to insert a blank row, column, or cell into the form.
<a href="#">Cn</a>	Use to define each column in a form.
<a href="#">CalcByRow</a>	Use to specify if the row calculation is used when a cell has an intersecting column calculation.
<a href="#">Cell_Link</a>	Use with <a href="#">Link</a> in a row definition to link to another data entry form.
<a href="#">CellText</a>	Use to specify whether the row or column accepts cell text input.



Table 7-1 (Cont.) Data Form Script Syntax

Script Syntax	Description
<a href="#">CustomHeader</a>	Use to specify custom header text to be displayed in place of the member label or description. Use in a row or column definition. You cannot use these keywords for Custom Headers: <ul style="list-style-type: none"> <li>• <code>&lt;pre&gt;</code></li> <li>• <code>&lt;textarea&gt;</code></li> <li>• <code>&lt;script&gt;</code></li> <li>• <code>&lt;javascript&gt;</code></li> <li>• <code>&lt;jscript&gt;</code></li> <li>• <code>&lt;vbs&gt;</code></li> <li>• <code>&lt;vbscript&gt;</code></li> <li>• strings such as <code>&lt;XonX=X&gt;</code>, where X = any string</li> </ul>
<a href="#">CustomHeaderStyle</a>	Use to assign custom style attributes to a row or column header.
<a href="#">DynamicPOV</a>	Deprecated. Do not use.
<a href="#">FormInputBoxLength</a>	Use to specify the input box width on the form.
<a href="#">FormNumDecimals</a>	Use to specify the number of decimal spaces for the form. This keyword overrides the decimal settings for the cell currency. Use <a href="#">NumDecimals</a> to override this setting for a row, column, or cell.
<a href="#">FormRowHeight</a>	Use to specify the height of all rows in the form.
<a href="#">FormScale</a>	Use to specify the scale for the form.
<a href="#">HeaderOption</a>	Use to specify how dimension headers display in the form. Show labels and/or descriptions, set style attributes, set maximum or fixed width.
<a href="#">Instructions</a>	Use to create instructions in HTML-formatted text and links.
<a href="#">LineItemDetailSinglePeriod</a>	Use to specify if line item detail displays for the selected cell only or for all periods.
<a href="#">Link</a>	Use with <a href="#">Cell_Link</a> to link to another data entry form.
<a href="#">MaxCells</a>	Use to specify the maximum number of cells for a data form.
<a href="#">MaxColsForSparseRetrievalMethod</a>	Use to optimize performance of sparse forms. Use with forms containing more than 10 columns.
<a href="#">NoSuppress</a>	Use to turn off suppression for one or more rows or columns. This setting overrides other suppression settings in the form: <a href="#">SuppressInvalidRows</a> , <a href="#">SuppressNoDataRows</a> , <a href="#">SuppressZeroRows</a> , <a href="#">SuppressInvalidCols</a> , <a href="#">SuppressNoDataCols</a> , <a href="#">SuppressZeroCols</a> .
<a href="#">NumDecimals</a>	Use to specify the number of decimal places for a row, column, or cell. This keyword overrides the decimal settings for the cell currency and the decimal setting for the form <a href="#">FormNumDecimals</a> .

Table 7-1 (Cont.) Data Form Script Syntax

Script Syntax	Description
<a href="#">OnDemandRules</a>	Use to specify which on-demand rules are available for the data form.
<a href="#">Override</a>	Use to specify a different POV or calculation for one or more rows or columns or to add style attributes or set the scale. Use in a row or column definition.
<a href="#">HideInPov</a>	Use to specify whether to hide the dimension in the POV.
<a href="#">POVOrder</a>	Use to specify the order of dimension names in the POV.
<a href="#">PrintNumDataColsPerPage</a>	Use to specify the number of columns to print on each page.
<a href="#">PrintNumRowsPerPage</a>	Use to specify the number of rows to print on each page.
<a href="#">PrintRepeatHeadersonAllPages</a>	Use to print headers on each page.
<a href="#">Rn</a>	Use to define each row in a form.
<a href="#">ReadOnly</a>	Use to specify read only rows, columns, or cells.
<a href="#">ReportDescription</a>	Use to specify the description for the form. The description cannot contain an ampersand (&).
<a href="#">ReportLabel</a>	Use to specify the label for the form. These characters are not supported for Data Form labels: Ampersand (&), asterisk (*), backslash (\), colon (:), comma (,), curly brackets ({}), double quotation marks (""), forward slash (/), less than and greater than (< >), number sign (#), parentheses ( ), period (.), pipe ( ), plus (+), question mark (?), semi-colon (;), and trailing underscore (_).
<a href="#">ReportSecurityClass</a>	Use to specify the security class for the form.
<a href="#">ReportType</a>	Use to set the form type. The value must be set to WebForm.
<a href="#">RowHeaderPct</a>	Use to resize the row header width in reference to the total width of the form.
<a href="#">SCalc</a>	Use to specify server-side calculations for a row, column, or cell.
<a href="#">Scale</a>	Use to specify scale for a row, column, or cell. Valid values are -12 to 12. This setting overrides the form scale setting. See <a href="#">FormScale</a> .
<a href="#">SelectablePOVList</a>	Use to specify the selectable dimension members in the form.
<a href="#">ShowDescriptions</a>	Use to show descriptions for dimension members.
<a href="#">ShowLabels</a>	Use to show labels for dimension members.
<a href="#">String</a>	Use to add a text string to a column, row, or cell.
<a href="#">Style</a>	Use to specify the style attributes for a row, column, cell, or dimension header.
<a href="#">SuppressColHeaderRepeats</a>	Use to prevent repeated column headers from displaying.

**Table 7-1 (Cont.) Data Form Script Syntax**

Script Syntax	Description
<a href="#">SuppressInvalidCols</a>	Use to prevent invalid cells in from displaying in columns.
<a href="#">SuppressInvalidRows</a>	Use to prevent invalid cells from displaying in rows.
<a href="#">SuppressNoDataCols</a>	Use to prevent columns with no data from displaying.
<a href="#">SuppressNoDataRows</a>	Use to prevent rows with no data from displaying.
<a href="#">SuppressRowHeaderRepeats</a>	Use to prevent repeated row headers from displaying.
<a href="#">SuppressZeroCols</a>	Use to prevent columns with zeros from displaying.
<a href="#">SuppressZeroRows</a>	Use to prevent rows with zeros from displaying.

## AddMember

Use this option within a Row definition to add an icon to the form which, when clicked, enables users to select members to add to the form. The new members are added to the form and the user can then enter data for these members.

You can use the AddMember option with the Account, ICP, and Custom dimensions.



### Note:

NoData cells can be suppressed by using the [SuppressNoDataRows](#) keyword or selecting the applicable check box on the form.

### Syntax

```
AddMember:MemberList
```

Replace *MemberList* with the list name from which users will be able to add members to the form.



### Caution:

You can reference only one member list per AddMember line.

To use this option, two separate row definitions are needed:

- A summary row that displays totals for the member list and has the AddMember option.

 **Note:**

The summary row cannot be suppressed.

- A list row that has the same POV specified in the summary row, except that it has a list for the dimension to which members are being added. The list must contain at least the members from the summary row list.

 **Note:**

The list row can be placed before or after the summary row.

**Example**

```
R1=A#SalesInterco.I{[Base]}
```

```
R2=A#SalesInterco.I#[ICP Entities],  
AddMember:I{[Base]}
```

In this example, Row 1 specifies the [Base] member list for the Intercompany Partner dimension. Row 2 specifies the total for the member list and the member list to be used with the AddMember icon.

 **Note:**

You can use the AddMember option in multiple rows in a form, but each summary row needs its own list row.

## BackgroundPOV

Use this keyword to specify the background dimension members for the form. Dimensions not specified in the Background POV are considered dynamic and are taken from the user's point of view when the form is opened.

Users need full security access to the member hierarchy to be able to work with all the members in the hierarchy. For example, if you want users to access all Custom 4 members, you must enable access to the parent entity, in this case, CustomTop.

 **Note:**

If a member in the Background POV is not valid based on the [SelectablePOVList](#) for the dimension, the system defaults to the first member of the list.

## Syntax

```
BackgroundPOV=S#Scenario.W#View.E#Parent.Entity.V#Value.A#Account.I#ICP.
C1#Custom1.C2#Custom2.C3#Custom3.C4#Custom4
```

**Table 7-2 Syntax for BackgroundPOV Keyword**

Parameter	Description
Scenario	Name of a valid Scenario member.
View	A valid view.
Parent	Name of a valid Parent member. This parameter is optional.
Entity	Name of a valid Entity member.
Value	Name of a valid Value member.
Account	Name of a valid Account member.
ICP	Name of a valid ICP member.
Custom	Name of valid Custom members.

 **Note:**

You do not need to specify all dimension members in the Background POV. For dimensions for which you do not specify a member, the system uses the dimension member from the user's point of view.

## Example

```
BackgroundPOV=S#Actual.Y#2014.P#January.W#Periodic.V#<Entity
Currency>.A#Sales.I#[ICPNone].C4#[None]
```

## Blank

Use this option to specify a blank row, column or cell in the form. Use this option within a row or column definition or within a cell override definition. The blank row, column, or cell is empty and is read only, and the context menu is disabled on the cell because it contains no data or POV. The Blank option is considered a server-side calculation, therefore it can be used anywhere that SCalc is used.

### Example

```
C4=Blank
```

## Cn

Use this keyword to define a column in the form. The keywords such as C1, C2, C3 provide the definition of each column in the specified order. The column identifier must begin with 1 and proceed in sequential order.

Dimension elements specified for the column override elements that are set in the [BackgroundPOV](#) or [SelectablePOVList](#). You can use member and system lists in column definitions.



### Note:

One list is allowed per column.

You can use these values and options within a column definition:

- [Blank](#)
- [Calc1](#)
- [CellText](#)
- [CustomHeader](#)
- [NumDecimals](#)
- [Override](#)
- [ReadOnly](#)
- [SCalc](#)
- [Scale](#)
- [String](#)
- [Style](#)

### Syntax

```
Cn=CalcExpression
```

```
Cn=POVExpression
```

**Table 7-3 Syntax for Columns Keyword**

Parameter	Description
n	The column number.
CalcExpression	Use SCalc, Blank, or String. See <a href="#">SCalc</a> , <a href="#">Blank</a> , and <a href="#">String</a> .
POVExpression	A valid dimension intersection or member list.

**Example**

```
C1=S#Actual.P#July,CustomHeader:Actual_July
```

```
C2=S#Actual.P#August,(Override 2,3, P#July)
```

```
C3=S#Budget.P#September
```

```
C4=SCalc(col(1)+col(3)),numdecimals:4,scale:1,readonly
```

```
C5=Blank
```

```
C6=S#Budget.P#October,Style:font-style:bold
```

```
C7=C1{TotalProducts.[Hierarchy]}
```

## Calc1

This value is Deprecated. If you have forms that use this keyword, you must manually edit them to use [SCalc](#).

## CalcByRow

A keyword used to change the default calculation order (columns first) to rows first. This keyword applies to the entire form.

**Syntax**

```
CalcByRow=Boolean
```

Where *Boolean* is True if row calculations are used; False if column calculations are used.

## Example

```
ReportType=WebForm
```

```
ReportLabel=CalcByCol
```

```
ReportDescription=Demonstrate CalcByRow
```

```
BackgroundPOV=S#Actual.Y#2014.P#January.w#<Scenario  
View>.E#UnitedStates.Connecticut.V#USD.A#Sales.I#[ICP  
NONE].C1#Golfballs.C2#Customer2.C3#[None].C4#[None]
```

```
C1=S#Actual
```

```
C2=S#Budget
```

```
C3=SCalc (Col (2) -Col (1) ) ,CustomHeader:Variance
```

```
C4=SCalc (Col (3) /Col (1) ) ,CustomHeader:Variance %
```

```
R1=A#Sales
```

```
R2=A#Purchases
```

```
R3=A#OtherCosts
```

```
R4=SCalc (Row (1) +Row (2) +Row (3) ) ,CustomHeader:Total
```

```
ShowLabels=True
```

```
CalcByRow=False
```

## Cell\_Link

Use this option to specify a link to another data form. Links are references in row definitions. Use with [Link](#). You can specify up to 64 links (Link1 - Link64). Links do not need to be numbered sequentially.



**Note:**

Linked form names are case-sensitive.

**Syntax**

```
Cell_Linkx
```

```
Linkx=FormName
```

Replace *x* with the number to assign to the link and replace *FormName* with the form name to which to link.

**Example**

```
R1=A#Salaries, Cell_Link1
```

```
Link1=Dynamic
```

## CellText

Use this option to display cell text for a row, column, or cell. The first 69 characters of the cell text entry are considered the title of the entry.

**Note:**

When you extract a data form containing cell text, only the title (first 69 characters) of a cell text entry is extracted.

**Syntax**

```
CellText:<cell text label>
```

**Example**

```
R1=A#Salaries, CellText:[Default]
```

```
R2=A#Salaries, CellText:Validation
```

## CustomHeader

Use this option to specify a custom header for a column or row. This is useful when you have a calculated column or row and want to hide the specific formula and replace it with a

description, such as Variance. If you have nested dimensions in rows or columns, the custom header applies to the entire header in the row or column, not just to the dimension.

You can use a semicolon as a delimiter to specify custom headers for subsequent cells. For example, this syntax replaces the three dimension headers in the row with custom headers Scenario, Year, and Month:

```
R2=S#Actual.Y#2014.P#January,CustomHeader:Scenario;Year;Month
```

To replace some header cells but not others, use a period (.) to indicate that the original header should be displayed. You can also hide a header by leaving out the period. For example, the following syntax shows the original header for the first dimension, hides the header for the second dimension, and shows the original header for the third dimension.

```
R2=S#Actual.Y#2014.P#January,CustomHeader:.;;.
```

### Caution:

When the CustomHeader option is used with calculated rows or columns, you can only set the text for the first cell of the header.

### Syntax

```
CustomHeader:HeaderName
```

Replace *HeaderName* with the header to use.

### Note:

You cannot use a comma (,), colon (:), or ampersand (&) in the custom header.

### Example

This example sets the custom header for column 3 to Variance.

```
C3=Scalc(col(1)-col(2)),customheader:Variance
```

## CustomHeaderStyle

Use this option to assign custom style attributes to a row or column header. This is different from the Style option of the HeaderOption keyword in that it applies to a row or column header as a whole as opposed to a single dimension across all headers. When there is a conflict, the CustomHeaderStyle option will be used over HeaderOption:Style keyword. To mix the two styles instead of the CustomHeaderStyle being used, insert a semicolon before the CustomHeaderStyle option as in the example below. See [Style](#).

### Syntax

This option uses the standards supported by the W3C.

```
CustomHeaderStyle:Property:Value
```

### Example

```
C1=S#Actual,CustomHeaderStyle:font-style:italic;font-family:arial;font-size:12px;font-color:red
```

## DynamicPOV

This keyword is deprecated.

If you have a form that contains the `DynamicPOV` keyword, use these steps to account for this deprecation:

1. Remove all dimensions specified with the `DynamicPOV` keyword from the `BackgroundPOV` keyword.
2. Delete the `DynamicPOV` keyword.

## FormInputBoxLength

Use this keyword to specify the input box width. The default is 20 characters wide.



### Note:

This keyword does not determine the number of characters that can be entered into the input box.

### Syntax

```
FormInputBoxLength=InputLength
```

Replace *InputLength* with the number of characters for the input box width.

### Example

```
FormInputBoxLength=20
```

## FormNumDecimals

Use this keyword to specify the number of decimals for the entire form. If this keyword is specified, it overrides the number of decimals set for the cell. If this keyword is not specified, the number of decimals for the cell is used.

You can override a column, row, or cell decimal setting by using [NumDecimals](#).

## Syntax

```
FormNumDecimals=Decimals
```

Replace *Decimals* with a value from 0 to 9.



### Note:

When you use a value of 9, maximum precision is used and therefore, up to 14 digits are actually included after the decimal.

## Example

```
FormNumDecimals=0
```

## FormRowHeight

Use this keyword to specify the height of all rows in the form. The default is 16px.

## Syntax

```
FormRowHeight=Pixels px
```

Replace *Pixels* with the number of pixels for the row height.



### Caution:

You must include `px` after *Pixels*. If you omit `px`, rows and columns may not align correctly when the form is printed.

## Example

```
FormRowHeight=16 px
```

## FormScale

Use this keyword to specify the default scaling for the form.

If specified in a form, this keyword overrides the scale in the entity currency, otherwise the system uses the scaling defined for the currencies assigned to entities.

You can override the form scale setting by using the [Scale](#) option within row or column definitions or within a cell override.

### Syntax

```
FormScale=n
```

Replace *n* with a value from -12 to 12.

### Example

```
FormScale=0
```

## HeaderOption

Use this keyword to specify heading defaults for each dimension. The options currently supported are:

- Length - the maximum length for row headers. This can be a number or can be the word "Fixed" if you want the maximum and minimum length to be the same.



#### Note:

Row headers are truncated with ellipses if they are longer than the specified length.

- ShowDescription to display member descriptions
- ShowLabel to display member labels
- Style

### Syntax

```
HeaderOptionDimension=Length:n
```

```
HeaderOptionDimension=ShowDescription
```

```
HeaderOptionDimension=ShowLabel
```

```
HeaderOptionDimension=Style:Property:Value;Property:Value...
```

Replace *Dimension* with the row dimension name, *n* with the length for the header or "Fixed", *Property* with the style property and *Value* with the property value. See [Style](#).

### Example

```
HeaderOptionPeriod=Length:4
```

```
HeaderOptionScenario=ShowDescription
```

```
HeaderOptionsAccount=ShowLabel,Style:font-style:italic
```

## HideInPov

Use this keyword to specify the HideInPOV option for each dimension. If this value is present, the dimension is not shown in the POV Bar of the data form. If this value is not found, the dimension is displayed in the POV Bar.

### Syntax

```
HideinPOV=Dimension
```

Dimension names can be either the short name or the long name, and are listed in a comma-separated list.



#### Note:

The dimension is ignored if it is already part of the selectable POV, since all selectable dimensions are shown in the POV bar.

### Example

In the following example, Scenario, Year, and Period are hidden in the POV bar when the data form is opened.

```
HideinPov=S,Y,P
```

## Instructions

Use this keyword to specify the instructions for form users. If you do not specify this keyword, the instructions window opens with this message: "There are no detailed instructions defined for this form".

### Syntax

```
Instructions=HTMLInstructions
```

Replace *HTMLInstructions* with HTML-formatted text and links.

### Example

Instructions=Please enter your cost center budgets for the year.Any questions, please contact the Budget Administrator.

## LineItemDetailSinglePeriod

Use this keyword to specify whether line item details are displayed for just the selected cell or for all input periods. The default is True, which displays details for only the selected cell.

### Syntax

```
LineItemDetailSinglePeriod=Boolean
```

Replace *Boolean* with True to display line item detail for the selected cell or False to show line item detail for all input periods.

### Example

```
LineItemDetailSinglePeriod=True
```

## Link

Use this keyword to specify a link to another data form. Links are references in row definitions. Use with [Cell\\_Link](#). You can specify up to 64 links (Link1 - Link64). Links do not need to be numbered sequentially.



### Note:

Linked form names are case-sensitive.

### Syntax

```
Cell_Linkx
```

```
Linkx=FormName
```

Replace *x* with the number to assign to the link and replace *FormName* with the form name to which to link.

### Example

```
R1=A#Salaries, Cell_Link1
```

```
Link1=Dynamic
```

## MaxCells

This keyword specifies the maximum number of cells allowed in the data form. If the form results in more than the MaxCells value, an error occurs. The default is 25000.

### Syntax

```
MaxCells=n
```

Replace *n* with the number of cells for the form.

#### Note:

The value is for the number of visible cells on the form including calculated cells. It does not include suppressed cells.

### Example

```
MaxCells=500
```

## MaxColsForSparseRetrievalMethod

#### Note:

This keyword is Deprecated. If you have a script that uses this keyword, the form loads correctly, however you cannot edit the value.

Use this keyword to specify the number of columns in the form to optimize performance of sparse data forms. You specify this keyword for forms that have more than 10 columns. If your form has 10 or fewer columns, the optimization occurs automatically.

### Syntax

```
MaxColsForSparseRetrievalMethod=n
```

Replace *n* with the number of columns in the form.

### Example

```
MaxColsForSparseRetrievalMethod=11
```



## NoSuppress

Use this option to turn off suppression for one or more rows or columns. Thus, regardless of the form suppression options, the row or column is displayed. Use this option within a row or column definition.

### Example

```
R4=A#Inventory,NoSuppress
```

## NumDecimals

Use this option to specify the number of decimal places to show for calculated or uncalculated rows or columns or in a cell override. If this keyword is specified, it overrides the number of decimals set for the cell or set by [FormNumDecimals](#).

### Syntax

```
NumDecimals:n
```

Replace *n* with a value from 0 to 9.



### Note:

When you use a value of 9, maximum precision is used and therefore, up to 14 digits are actually included after the decimal.

### Example

```
C4=A#Inventory,NumDecimals:1
```

## OnDemandRules

Use this option to specify which on-demand rules are available for the data form. Use a comma-separated list of rule names. See [Specifying On-Demand Rules for Data Forms](#).

### Example

```
OnDemandRules=Calculation,Tax,Tax2
```

## Override

Use this option to specify different POV dimension members, formula calculations, or text for one or more consecutive columns or rows or to change a style. Use this option within a row or column definition.

 **Note:**

To override cells that are not consecutive, you can enter the override in the Other field of the form options. You can enter multiple overrides by separating each override by a comma. The following example overrides three individual cells:

```
Override(1,1,string("455")),Override(3,3,string("23")),
Override(5,5,string("2234"))
```

You cannot use member lists with the Override option. You can apply multiple overrides to a cell, with the last value being used when there is a conflict. In most cases, you can also mix the overrides on a cell. For example, if an override on a row specifies a value for NumDecimals while an override on a column specifies a value for Scale, there is no conflict except if the same dimension is used. If you use a leading semicolon, you can mix values for Style.

You can also mix overrides where they intersect by including a semicolon after the Override keyword. Note that without the semicolon, the style defined for the row is used because row values supersede column values when they conflict in the form. To mix the overrides from the row and column definitions, you must include the semicolon on the row keyword.

**Syntax**

```
Override(StartCell,EndCell,Override)
```

Parameter	Description
StartCell	<p>An integer value representing the override starting point. If the override is defined for a row, this parameter indicates the starting column where the override is applied. In this example, the override starts with column 2:</p> <pre>R2=A#Sales, Override(2,3,A#SalesTP)</pre>
EndCell	<p>An integer value representing the override ending point. In the example above, the override ends with column 3.</p>
Override	<p>A POV. For example, to override with a different account, you specify A#newacct. To override with a different scenario and account, you specify A#newacct.S#newscenario.</p> <p>You can also use these values or options with a POV override or by themselves:</p> <ul style="list-style-type: none"> <li>• Blank</li> <li>• CellText</li> <li>• NumDecimals</li> <li>• ReadOnly</li> <li>• SCalc</li> <li>• Scale</li> <li>• String</li> <li>• Style</li> </ul>

### Example

In the following override example, the system overrides columns 2 and 3 for row 2 with January as the period and the PriorSales amount instead of the sales amount for the month:

```
R2=A#Sales, Override(2,3,Y#2014.P#January.A#PriorSales)
```

In the following override example, the system overrides columns 2 and 3 for row 2 with a formula calculation of the average of Sales1, Sales2, and Sales3:

```
R2=A#Sales, Override(2,3,S Calc((A#Sales1+A#Sales2+A#Sales3)/3)
```

In the following override example, the system overrides columns 2 and 3 for row 2 with the read only option.

```
R2=A#Sales, Override(2,3,readonly)
```

## POVOrder

Use this keyword to specify the order of dimension names in the POV. Dimension names can be either the short name or the long name and are listed in a comma-separated list in the order in which they should display. Any dimension not specified on the list is appended in order by dimension number to the end of the list.

### Example

In the following example, the dimension order on the POV bar is Account, Period, Year, Scenario.



#### Note:

If the Account dimension was specified as hidden, Period would display first on the POV bar.

```
POVOrder=A, P, Y, S
```

## PrintNumDataColsPerPage

Use this keyword to specify the number of columns to print on each page. The default is 6.

Users can override this setting in the printer-friendly display of the form.



#### Note:

When setting the value for this keyword, you should also take into account the specifics of the form, such as the row height, and the printer and printer settings used, including resolution and orientation.

**Syntax**

```
PrintNumDataColsPerPage=n
```

Replace *n* with the number of columns to print per page.

**Example**

```
PrintNumColsPerPage=6
```

## PrintNumRowsPerPage

Use this keyword to specify the number of rows to print on each page. The default is 20.

Users can override this setting in the printer-friendly display of the form.

**Note:**

When setting the value for this keyword, you should also take into account the specifics of the form, such as the row height, and the printer and printer settings used, including resolution and orientation.

**Syntax**

```
PrintNumRowsPerPage=n
```

Replace *n* with the number of rows to print per page.

**Example**

```
PrintNumRowsPerPage=20
```

## PrintRepeatHeadersonAllPages

Use this keyword to specify whether to print headers on all pages. The default is True.

Users can override this setting in the printer-friendly display of the form.

**Syntax**

```
PrintRepeatHeadersonAllPages=Boolean
```

Replace *Boolean* with True or False.

**Example**

```
PrintRepeatHeadersonAllPages=True
```

## Rn

Use this keyword to define a row in the form. The keywords such as R1, R2, R3 provide the definition of each row in the specified order. The row identifier must begin with 1 and proceed in sequential order. You can use member and system lists in row definitions.

**Note:**

You can use multiple lists in a row.

Dimension elements specified for the row override elements that are set in the [BackgroundPOV](#) or [SelectablePOVList](#).

You can use these values and options within a row definition:

- [AddMember](#)
- [Blank](#)
- [Cell\\_Link](#)
- [CellText](#)
- [CustomHeader](#)
- [NoSuppress](#)
- [NumDecimals](#)
- [Override](#)
- [ReadOnly](#)
- [SCalc](#)
- [Scale](#)
- [String](#)
- [Style](#)

**Syntax**

*Rn=CalcExpression*

*Rn=POVExpression*

**Table 7-4 Syntax for Rows Keyword**

Parameter	Description
n	The row number.

**Table 7-4 (Cont.) Syntax for Rows Keyword**

Parameter	Description
CalcExpression	Use SCalc, Blank, or String. See <a href="#">SCalc</a> , <a href="#">Blank</a> , and <a href="#">String</a> .
POVExpression	A valid dimension intersection or member list.

**Example**

```
R1=A#Sales.I#[ICP Entities],AddMember:I{[Base]}
```

```
R2=A#Sales.I{[Base]}
```

```
R3=A#HeadCount.I#[ICP None],NoSuppress
```

```
R4=A#Purchases.I#[ICP None],CustomHeader:ABC
```

```
R5=SCalc(Row(2)*100),numdecimals:1,scale:0
```

```
R6=A{OperatingIncome.[Descendants]}
```

## ReadOnly

Use this option to specify a read-only row, column, or cell in the form. Use this option within a row or column definition or within a cell override definition. The read-only cell is similar to all other cells but you cannot edit its contents. You can modify the style of a read-only row, column, or cell to differentiate it from editable rows, columns, and cells. See [Style](#).

**Note:**

When you export to Excel, the read-only formatting is maintained.

**Example**

```
C4=S#Actual.Y#2014,ReadOnly
```

## ReportDescription

Use this keyword to specify the form description.

## Syntax

```
ReportDescription=Description
```

Replace *Description* with a description for the form. The description can contain a maximum of 255 characters.

## Example

```
ReportDescription=Intercompany Detail
```

# ReportLabel

Use this keyword specify the form name. This keyword is required.

## Syntax

```
ReportLabel=Label
```

Replace *Label* with the form name. The name can contain a maximum of 40 characters. You cannot use these characters in the name:

- Asterisk (\*)
- At sign (@)
- Backslash (\)
- Colon (:)
- Comma (,)
- Curly brackets ({})
- Forward slash (/)
- Less-than and greater-than signs (<)
- Number sign (#)
- Parentheses ( )
- Period (.)
- Pipe (|)
- Plus sign (+)
- Question mark (?)
- Quotation marks ""
- Semicolon (;)

 **Note:**

You can use an underscore ( `_` ) in the report label but it must be used between two characters. It cannot be used alone as the label name and it cannot be used at the end of a label name.

**Example**

```
ReportLabel=ICP Detail
```

## ReportSecurityClass

Use this keyword to specify the security class assigned to the form. The default is [Default].

**Syntax**

```
ReportSecurityClass=SecurityClass
```

Replace *SecurityClass* with the name of a valid security class.

**Example**

```
ReportSecurityClass=Corporate
```

## ReportType

Use this keyword to specify the report type. This keyword is required and the value must be set to WebForm for the file to be loaded as a data form script.

**Syntax**

```
ReportType=WebForm
```

## RowHeaderPct

Use this option to resize the row header width in reference with the total width of the form. This is applicable only when the actual width of the row header exceeds the specified percentage.

For example, if the actual row header width is 25% of the total form's width, and the specified value is 40%, the row header would not increase to 40%, as it could display the content in 25%. However, if the specified value is 10%, the row header width would decrease from the actual width of 25% to fit in the specified 10% width with a scroll bar. If a row header has multiple columns, the system adds the width of all the columns, and compares it with the total width of the form.

**Syntax**

```
RowHeaderPct:n
```

Replace *n* with a value from 1 to 100.



## Example

```
RowHeaderPct=30
```

## SCalc

You can use the `SCalc` function to create, in columns and rows, custom formulas that use standard mathematical operators. Use this value within a row or column definition or within a cell override definition. The specified calculations are performed on the application server. The following example subtracts column 2 from column 1:

```
SCalc(col(1)-col(2))
```

You can also use this value to create text within the form.

## Syntax

```
SCalc(<expression> [<operator> <expression>])
```



### Note:

You can include multiple pairs of `[<operator> <expression>]` in an `SCalc` calculation.

Parameter	Description
<code>operator</code>	The mathematical operator for the calculation. These operators are supported: + - * /
<code>expression</code>	The values in the calculation. In addition to numeric values, you can include cell references, row references, column references, and various other types of items.

You can use these types of items in `SCalc` calculations:

- References to dimension members. This example references the Account dimension members `Purchases` and `OtherCosts`: `R6=SCalc(((A#Purchases)-(A#OtherCosts))*100)`
- Cell references, using the syntax `Cell(rowIndex, columnIndex)`. This example refers to the cell in the fourth row of the second column in the form: `R1=SCalc(Cell(4,2))`
- Row references, using the syntax `Row(rowIndex)`. This example divides row 4 by row 2: `R3=SCalc(Row(4)/Row(2))`



### Note:

For rows or columns that contain member lists, the calculation occurs on the total for the members of the list.

- Column references, using the syntax `Col (columnIndex)`. This example adds column 1 and column 3: `C4=SCalc(Col(1)+Col(3))`
- Nested formulas, using parentheses to nest.
- Forward references to cells with SCalc calculations

 **Note:**

SCalc calculations are not performed until the data is saved and the calculated results are not displayed on the form until the form is refreshed.

An SCalc row or column can reference another SCalc row or column in its calculation, however you cannot forward reference in a SCalc row or column for another SCalc row or column. For example, this SCalc forward reference is allowed:

```
C1=A#Sales
C2=A#COGS
C3=SCalc(Col(1)-Col(2))
C4=SCalc(Col(3)/Col(1)*100)
```

## Scale

Use this option to specify the scale for uncalculated columns, rows, or cells.

### Syntax

```
Scale:n
```

Replace *n* with a value from -12 to 12.

### Example

```
C4=A#Inventory,Scale:2
```

## SelectablePOVList

Use this keyword to specify the members of a dimension that can be selected by users.

The initial value for a selectable dimension comes from the Background POV if one is specified for the dimension; otherwise, the initial value comes from the user's point of view. If the initial value is invalid for the list, the system defaults to the first member of the list.

When the user selects a new member, the selected member becomes part of the user's point of view.

## Syntax

```
SelectablePOVList=Dimension{MemberList}
```

```
SelectablePOVList=Dimension{ [SystemList] }
```

```
SelectablePOVList=Dimension{Parent. [SystemList] }
```

**Table 7-5 Syntax for SelectablePOVList Keyword**

Parameter	Description
<code>Dimension</code>	<p>One of these characters to represent the dimension that is selectable:</p> <ul style="list-style-type: none"> <li>• S for Scenario</li> <li>• W for View</li> <li>• E for Entity</li> <li>• V for Value</li> <li>• P for Period</li> <li>• A for Account</li> <li>• I for ICP</li> <li>• C for Custom</li> </ul>
<code>Parent</code>	A valid parent for the dimension.
<code>MemberList</code>	Name of a valid member list.
<code>SystemList</code>	Name of a valid system list.

## Example

```
SelectablePOVList=S{ActualBudget}.Y{Years}.P{Months}.E{[Hierarchy]}.  
C1{ProductsAllocate}.C2{Customers}.C3{AllChannels}
```

## ShowDescriptions

Use this keyword to specify whether descriptions are displayed for the dimension elements in the form. The default is False.

### Note:

If no description exists, the label is displayed. If there is no description for an entity, only the entity label is displayed; the parent label is not included.

### Syntax

```
ShowDescriptions=Boolean
```

Replace *Boolean* with True or False.

### Example

```
ShowDescriptions=True
```

## ShowLabels

Use this keyword to specify whether labels are displayed for the dimension elements in the form. The default is True.

If [ShowLabels](#) and [ShowDescriptions](#) are set to True, then labels and descriptions are separated by hyphens. If both are set to False, the system displays labels.

### Syntax

```
ShowLabels=Boolean
```

Replace *Boolean* with True or False.

### Example

```
ShowLabels=False
```

## String

Use this option to specify a text string in a column, row, or cell. Use this option within a row or column definition or within a cell override definition. The String option is considered a server-side calculation, therefore it can be used anywhere that SCalc is used.

### Example

```
C4=String ("Show this read-only string")
```

## Style

Use this option to specify the style attributes for a row, column, cell, custom header, or dimension header. Use this option within a row, column, cell override, custom header or header option definition.

### Caution:

Use care when applying style attributes to a form. The Style option is a powerful feature and, if used incorrectly, can significantly change the appearance of a form.

**Note:** When you export to Excel, the style formatting is maintained.

### Syntax

Each style property consists of a property name, colon, and value. The value may be one or more words, separated by spaces, for example:

```
Style:color:red;text-align:right
```

The Style option is interpreted by the browser, not by Oracle Hyperion Financial Management, so it is limited only by what the browser supports. The Style option uses the standards supported by the World Wide Web Consortium (W3C).

You can use the Style option keyword one time on a line and add multiple Property:Value pairs. For example:

```
C3=S#Actual.Y#@CUR(-1),Style:font-color:blue;background-color:red;font-weight:bold
```

The Style Example table lists some of the properties and potential values that can be used. For a full list, see the Property Index from the W3C.

**Table 7-6 Style Examples**

Property	Value
Color (foreground) Background-color	The color name or the standard hexadecimal RGB notation. For example: Style: color: red Style: color: rgb(255,0,0) Style: background-color: #ff0000 Style: background-color: yellow
Font-family	The font name. For example: Style: font-family: Arial
Font	The font style. For example: Style: font-style: italic You can combine up to six font properties in a Style value. For example: Style: font-style:italic;font-family:Arial;font-size:12px
Font-weight	The font weight. Values include demi-bold, demi-light, extra-bold, light, extra-light, demi-light. For example: Style: font-weight: extra-bold
Font-size	The font point size. For example: Style: font-size: 12px
Alignment	The text alignment. Values include left, right, center, and justify. For example:  Style: text-align: center

You can combine styles at row and column intersections by including, for a row, a semicolon as the first character in the Style text box or Cust Header Style text box. Note that without the semicolon after the Style keyword, the style defined for the row is used because row values supersede column values when they conflict in the form.

Each cell in the grid displays only its right and bottom borders, so that a single pixel separates adjoining cells. You can use Style to change the color or set the line to dashed for the right or bottom borders. However, if you enable the top or left border, it will be in addition to the bottom border of the above cell and the right border of the cell to the left, respectively. This means you will have two borders. You can avoid double borders by turning off the adjacent border.

You must also set the CustomHeaderStyle of an axis to keep the headers aligned with the grid.

### Example

In the following example, the styles from Row 1 (pink background) and Column 1 (pink background and bold, green text) are combined by adding a semicolon after the Style keyword in the row definition. The example for [SuppressColHeaderRepeats](#) shows a sample data entry form if this syntax is used.

```
R1=Blank, CustomheaderStyle: font-weight: bold, CustomHeader: Balance sheet accounts, Style;; Background-color: pink
```

```
C1=S#Actual.Y#@CUR, Style: Background-color: pink; color: green; font-weight: bold
```

	<i>Actual</i>	<i>Budget</i>
<b>Balance sheet accounts</b>		
Cash		1,746,137.00
Short Term Receivables		9,951.32
Inventories		-10,746.57
Short Term Investments		2,831.47
Total Short Term Assets		1,748,173.21
Computers	<b>33,333.00</b>	13,382.68
Buildings		4,351.48
Office Equipment	<b>50,000.00</b>	-948.88
Transportation	<b>80,000.00</b>	4,719.12
Fixtures	<b>800,000.00</b>	931.97
Tangible Assets	<b>963,333.00</b>	22,436.37
Accumulated Depreciation	<b>22,222.00</b>	-87,002.34

## SuppressColHeaderRepeats

Use this keyword to specify whether to suppress repeated column headers. The default is True.

This example shows the data form where repeated column headers are suppressed. If suppression is turned off, the column header Actual would also be displayed above August.

	Actual	
	July	August
Sales	500,000.00	600,000.00
Purchases	250,000.00	300,000.00

### Syntax

```
SuppressColHeaderRepeats=Boolean
```

Replace *Boolean* with True or False.

### Example

```
SuppressColHeaderRepeats=False
```

## SuppressInvalidCols

Use this keyword to specify whether columns containing invalid cells are suppressed from the form. The default is False.



### Note:

Regardless of this setting, invalid columns are suppressed for non-administrator users.

### Syntax

```
SuppressInvalidCols=Boolean
```

Replace *Boolean* with True to suppress invalid columns or False to display invalid columns.

### Example

```
SuppressInvalidCols=False
```

## SuppressInvalidRows

Use this keyword to specify whether rows containing invalid cells are suppressed from the form. The default is False.



### Note:

Regardless of this setting, invalid rows are suppressed for non-administrator users.

**Syntax**

```
SuppressInvalidRows=Boolean
```

Replace *Boolean* with True to suppress invalid rows or False to display invalid rows.

**Example**

```
SuppressInvalidRows=False
```

## SuppressNoDataCols

Use this keyword to specify whether columns containing nodata cells are suppressed from the form. The default is False.

**Syntax**

```
SuppressNoDataCols=Boolean
```

Replace *Boolean* with True to suppress columns with no data or False to display columns with no data.

**Example**

```
SuppressNoDataCols=False
```

## SuppressNoDataRows

Use this keyword to specify whether rows containing nodata cells are suppressed from the form. The default is False.

**Syntax**

```
SuppressNoDataRows=Boolean
```

Replace *Boolean* with True to suppress rows with no data or False to display rows with no data.

**Example**

```
SuppressNoDataRows=False
```

## SuppressRowHeaderRepeats

Use this keyword to specify whether to suppress repeated row headers. The default is True.

**Syntax**

```
SuppressRowHeaderRepeats=Boolean
```



Replace *Boolean* with True or False.

### Example

```
SuppressRowHeaderRepeats=False
```

## SuppressZeroCols

Use this keyword to specify whether columns containing zeros are suppressed from the form. The default is False.

### Syntax

```
SuppressZeroCols=Boolean
```

Replace *Boolean* with True to suppress columns with zeros or False to display columns with zeros.

### Example

```
SuppressZeroCols=False
```

## SuppressZeroRows

Use this keyword to specify whether rows containing zeros are suppressed from the form. The default is False.

### Syntax

```
SuppressZeroRows=Boolean
```

Replace *Boolean* with True to suppress rows with zeros or False to display rows with zeros.

### Example

```
SuppressZeroRows=False
```

## Using Relative Time Periods

For rows and columns, you can use these keywords to select a period relative to the current POV:

- @Cur
- @First
- @Last
- @Next
- @Prev

 **Note:**

Year and period values are not case sensitive.

For example, if the period selected in the POV is January, the column or row that is created using this formula displays data for February:

P#CUR+1

For the system to apply the relative year when the period extends past the current year, or if you need to refer to a prior year, you must specify Y#@Cur as part of the row or column definition.

For example, you need to specify C1=Y#@Cur.P#@Cur(+1) to return the correct year if the current period is the last period. If the current Point of View is 2013 December: C1=Y#@Cur.P#@Cur(+1) returns 2014 January.

If the current Point of View is 2013 December: C2=P#@Cur(+1) returns 2014 January (if January is the first period in the year, and December is the last). Therefore, for the system to correctly reflect the year information, you specify the year in the column definition. For example, C2=Y#@Cur.P#@Cur(+1)

The @CUR function can be used for other dimensions to retrieve the current POV. However, @CUR for the other dimensions does not support using an offset, for example, +2, since the other dimensions are not chronological. For example, if the current POV had the Actual scenario, a valid example for a column would be to use C1=S#@CUR to return "Actual". If you changed the POV scenario to Budget, this same column definition would return "Budget".

## Order of Precedence for Conflicting Attributes

There are multiple ways to set the number of decimals, scale, and POV for data forms. For example, you can set the number of decimals:

- For the form - FormNumDecimals=3
- For a column - C4=A#Inventory, NumDecimals:1
- For a row - R4=A#Sales, NumDecimals:2
- For a cell override - R2=S#Actual.P#August, Override(2,7,P#July, NumDecimals:4)

In some instances, the setting for one of these attributes may intersect with a conflicting setting for the same attribute. For example, a column may have a scale setting of 1, while an intersecting row may have a scale setting of 2. It is important to understand that the value that is applied for the number of decimals and scale attributes is based on this order of precedence:

- Cell Override
- Row
- Column
- Form
- Default - For number of decimals, the default comes from the account. For scale, the default is taken from the entity currency.

 **Note:**

In the POV, the value for each dimension is independently resolved. For example, the Account dimension might be set at the form level and the Scenario dimension at the row level.

The system resolves conflicting attributes according to the order of precedence. For example, if the number of decimals attribute is defined on a cell override, this value is used instead of the form, row, column or default.

Similarly, if a cell has conflicting data and calculations specified, this order of precedence is applied:

- SCalc
- Data

 **Note:**

SCalc formulas in a form are evaluated in a left to right, top to bottom order. For example, all of the columns in row 1 are evaluated sequentially, then all of the columns for row 2 are evaluated. An SCalc formula can refer to another SCalc cell that precedes it in the evaluation order.

## Editing Data Forms

You can edit a data form using the Form Designer. You can only edit one data form at a time.

 **Note:**

After you edit a data form, you must reload the updated script to the application to see the changes that you made.

To edit data forms:

1. Select **Consolidation**, and then **Documents**.
2. Open a data form.
3. Click **Edit**, or select **Actions**, and then **Edit**.
4. Edit the form as needed.
5. Click **Save**.

## Loading Data Forms

After you create data form scripts, you load them into an application.

 **Note:**

Oracle recommends that you add Oracle Hyperion Financial Management to the exceptions for your Web pop-up blocker. When you perform some tasks such as loading data, a status window pops up showing the task status. If you have a pop-up blocker enabled on your computer, the status window is not displayed.

To load data forms:

1. Select **Consolidation**, then **Load**, and then **Documents**.
2. From **Document Type**, select **Data Form**.
3. Enter the file name or click **Browse** to locate the file.

 **Note:**

By default, data form scripts use the WDF file extension. The load process accepts other file extensions such as TXT or CSV, however, Oracle recommends that you use the WDF file extension.

4. **Optional:** Perform one of these steps:
  - To override the security class specified in the file being loaded, select **Override Security Class**, then select the security class from **Security Class**.
  - To make this form available only to you, select **Private**. If you select this option, the **Override** option and **Security Class** list are not available.
5. Repeat these steps until you have added the forms to load.
6. **Optional:** Select **Overwrite Existing Documents** if you are updating data forms.
7. Click **Load**.

 **Note:**

All data forms that you load must include a valid ReportLabel. If you selected to load multiple forms and one of the forms has an invalid ReportLabel, none of the selected forms are loaded.

## Extracting Data Forms

You can extract data form scripts from an application. Extracting the script does not delete the script from the folder or from the application. It only extracts the contents of the script to a location you select.

After you extract a data form script, you can modify it and reload it to the application. You can also use the model for a new script.



**Note:**

If you are reloading a script into an application, you must select the **Overwrite Existing Documents** option to replace the old file in the application.

To extract data forms:

1. Open the application.
2. Select **Consolidation**, then **Documents**.
3. Click **Extract Documents**, or select **Actions**, and then **Extract Documents**.
4. Enter the file name or click **Browse** to locate the file.
5. Click **Extract**.
6. Click **Save** and specify the location to which to save the file.

## Deleting Data Forms

To delete data forms, you must be an administrator with the security role of Manage Data Entry Forms. If you want to delete a folder, you must first delete any data forms that it contains.

To delete data forms:

1. Open the application.
2. Select **Consolidation**, then **Documents**, and then **Data Forms**.
3. Select a data form, then click **Delete**, or select **Actions**, and then **Delete**.

# 8

## Extracting Data to a Database

Use the procedures in this chapter to extract data to a database.



### Note:

To extract data as flat files, see the Extracting Data section in the *Oracle Hyperion Financial Management User's Guide*.

You can extract data and use an Oracle Essbase database to analyze the data and produce reports. You use a star schema to send data to an Essbase database. You can also use a star schema with third-party products.



### Note:

You create the database after you export data to the star schema. To create a database, see the database documentation for your release.

Before you extract data to a database, you must set up a data source name (DSN) for the database to store star schemas. To store star schemas in multiple databases, you can create a DSN for each database. For instructions, see the *Oracle Enterprise Performance Management System Installation and Configuration Guide*.

To extract data to a database, you must be assigned the Administrator or Extended Analytics security role.

If you are not assigned the Extended Analytics security role, these restrictions apply for extracting data:

- You can extract only base level data for ICP and all Custom dimensions.
- You can select only one Scenario and one Year dimension.
- You can select only the Periods that are in the Scenario's default frequency.

## Configuring a Data Source Name (DSN)

To use the Extract Data to Database feature, you must configure a data source name (DSN) to provide a connection between the database server and application server. The DSN specifies the database server name and other database-related information, such as the username and password of a user with full access rights to the database server.

To create a DSN:

1. Select **Navigate**, then **Administer**, and then **Consolidation Administration**.
2. From **Admin Tasks**, double-click **Configure DSN**.

3. Click **Create Data Source**.
4. Enter a Data Source Name for the destination database.
5. From the **Database Type** drop-down list, select a database type: **Oracle**, **SQL Server**, or **DB2**.
6. Enter a username and password.

 **Note:**

Ensure that the user is granted privileges to create, update, and delete tables.

7. Enter the server **Host** name.
8. Enter the **Port** number.
9. **Optional:** Enter a default tablespace or filegroup name.
10. **Optional:** Enter an index tablespace or filegroup name.
11. Click Test **Connection**.
12. When the system displays the "Successful" confirmation, click **Save** to save the DSN.
13. Click **Refresh** to display the updated DSN information.

After you have created and saved a connection, it is displayed on the Configure DSN page. If you need to edit it at a later time, you can edit it by clicking Edit DSN, or you can delete it when it is no longer needed.

## Star Schemas

You can create multiple star schemas per application. The table names in each star schema begin with a prefix that you specify. You can select multiple members in all dimensions to create the star schema that best reflects the information to export.

 **Note:**

Cell text and line item detail are not exported to the star schema.

The data combinations in the star schema are created based on the dimension members that you select to export. The more dimension members selected, the more possible data combinations that need to be created in the star schema, and the more time needed to complete the export process. You can calculate the number of data combinations by multiplying the number of members selected for each dimension.

 **Caution:**

Do not select to export all members from every dimension; select segments of data to export. Depending on the application size, the number of data combinations, and the amount of time to complete the export time could be excessive.

For example, you can export this data:

- Scenario - Actual
- Year - 2014
- Period - January
- View - Periodic
- Entity - Regional, United States, Florida, Connecticut
- Value - USD
- Account - Gross Margin, Sales, Total Costs
- ICP - [None]
- C1 - Balls, Tennis Balls, Golf Balls
- C2 - All Customers, Customer2, Customer3, Customer4, Customer5
- C3 - [None]
- C4 - [None]

The star schema that is created can then be used by Oracle Essbase Integration Services to create one or many data cubes to reflect the audience that needs to see and use the information. The star schema contains 180 data combinations for these members ( $1 * 1 * 1 * 1 * 4 * 1 * 3 * 1 * 3 * 5 * 1 * 1 = 180$ ).

 **Note:**

When you export metadata to Oracle Essbase through Integration Services, the Extract Data to a Database feature uses metadata in a different order than it displays in the Oracle Hyperion Financial Management application.

You can extract only local currency data if you prefer. To extract only local currency data, use the Entity Currency member from the Value dimension in the Point of View. Entity Currency acts as placeholder for the currency and extracts the default currency for each entity that is selected.

If the selected Scenario is YTD, the data extract extracts periodic derived data. For example, suppose the Scenario is ZeroView=YTD. There is a value of 100 in an expense account in the first period. In the second period, no data is reported for that account. The derived periodic value for the second period is -100, forcing the YTD amount to 0. If Extract Data to Database is run for this Point of View on a periodic basis, for the first three periods, the values are 100, -100, and 0, respectively.

## Star Schema Formats

You select one of these extract format options when you create a star schema:

- Standard
- Metadata Only
- Selected Metadata Only
- Essbase
- Data Warehouse



The extract format option that you select determines the schema format used. Each schema format generates a different set of tables. These schema formats are available:

- Standard Essbase Schema - this schema is used for the Standard, Metadata Only, and Selected Metadata Only extract format types.
- SQL and Essbase Schema - this schema is used for the Essbase extract format type.
- Warehouse Normalized Hierarchy Schema - this schema is used for the Data Warehouse extract format type.

## Prefix Tables

For each schema format, the system creates a *PREFIX\_FACT* table that contains keys to the dimension tables and one data field. The system also creates *PREFIX\_DIMENSION* tables, and the *HFM\_EA\_EXTRACT* table to track extract timestamps for metadata. The system creates two tables for the Entity dimension: *PREFIX\_ENTITY* and *PREFIX\_PARENT*.

For the SQL and Essbase Aggregation Schema format, the system also creates *PREFIX\_DIMENSION\_BASE* tables.

For the Warehouse Normalized Hierarchy Schema format, the system creates *PREFIX\_DIMENSION\_PARENT* tables.



### Note:

Base and Parent tables are not created for the View, Year, and Parent dimensions.

For example, if the Relational Table Prefix is DEMO, the system creates these tables for the Essbase format:

- HFM\_EA\_EXTRACT1
- DEMO\_FACT
- DEMO\_YEAR
- DEMO\_VIEW
- DEMO\_PARENT
- DEMO\_SCENARIO and DEMO\_SCENARIO\_BASE
- DEMO\_PERIOD and DEMO\_PERIOD\_BASE
- DEMO\_VALUE and DEMO\_VALUE\_BASE
- DEMO\_ENTITY and DEMO\_ENTITY\_BASE
- DEMO\_ICP and DEMO\_ICP\_BASE
- DEMO\_ACCOUNT and DEMO\_ACCOUNT\_BASE
- DEMO\_CUSTOM and DEMO\_CUSTOM\_BASE

## Creating and Exporting Data to a Star Schema

You can create a star schema by specifying the prefix that identifies the tables for the schema and the dimension members of the cells to be exported. Before you export the data, make sure that the application data is consolidated.

 **Caution:**

Do not use the Oracle Hyperion Financial Management database as the destination database for the data export.

**Note:** For Oracle database users: The extracted text data is stored in NVARCHAR(Unicode) format. Use the Oracle "translate" command in SELECT statements to convert the text from Unicode to ANSI format. For example, SELECT translate (LABEL using CHAR\_CS) as LABEL FROM DEMO\_ACCOUNT.

If you are extracting a large amount of data, make sure that you have adequate disk space for the extract.

Oracle recommends that you add Financial Management to the exceptions for your Web pop-up blocker. When you perform some tasks such as loading data, a status window pops up showing the task status. If you have a pop-up blocker enabled on your computer, the status window is not displayed.

To create a star schema and extract Financial Management data:

1. Select **Consolidation**, then **Extract**, and then **Data**.
2. Do one of these tasks:
  - If you saved POV dimensions in a template, from **Template**, select the template.
  - If you are not using a template, select the POV dimension members to export.
3. From the **Extract Destination** section, for the **Type** option, select **Database**.

 **Note:**

This option is only available if you are assigned the Administrator or Extended Analytics security role.

4. From **Extract Format**, select an option:
  - **Standard**
  - **Metadata Only** - extracts metadata only
  - **Selected Metadata Only** - extracts only the metadata for the selected dimension members.
  - **Essbase**
  - **Data Warehouse**
5. From **Options**, select one or more options:
  - **Extract Dynamic Accounts**
  - **Calculated Data**
  - **Derived Data**
6. From **Line Item Details**, select an option:
  - **Total Summary for Cell**
  - **None**

7. From **Schema Actions**, select **Create Star Schema**.
8. From **Destination Database (DSN)**, select the database to which you are exporting the star schema.

 **Note:**

Do not use the same Financial Management database from which you are exporting data as the destination database.

9. For **Relational Table Prefix**, enter a prefix to identify the tables for the star schema, or use the default application name prefix.

 **Note:**

The prefix can contain up to 10 alphanumeric characters and must start with a letter. It cannot contain an underscore. The default prefix is the application name.

10. Click **Extract**.

## Updating a Star Schema

You can export data to a previously defined star schema. When you update a star schema, you can specify different dimension members.

 **Note:**

When you update a star schema, the system updates the Fact table.

Oracle recommends that you add Oracle Hyperion Financial Management to the exceptions for your Web pop-up blocker. When you perform some tasks such as loading data, a status window pops up showing the task status. If you have a pop-up blocker enabled on your computer, the status window is not displayed.

To update a star schema:

1. Select **Consolidation**, then **Extract**, and then **Data**.
2. Do one of these tasks:
  - If you saved POV dimensions in a template, from **Template**, select the template.
  - If you are not using a template, select the POV dimension members to export.
3. From **Extract Destination Type**, select **Database**.

 **Note:**

This option is only available if you are assigned the Administrator or Extended Analytics security role.

4. From **Extract Format**, select an option:
  - **Standard**
  - **Metadata Only** - extracts metadata only
  - **Selected Metadata Only** - extracts only the metadata for the selected dimension members.
  - **Essbase**
  - **Data Warehouse**
5. From **Options**, select one or more options:
  - **Extract Dynamic Accounts**
  - **Calculated Data**
  - **Derived Data**
6. From **Line Item Details**, select an option:
  - **Total Summary for Cell**
  - **None**
7. From **Schema Actions**, select **Update Star Schema**.
8. From **Destination Database (DSN)**, select the database to which you are exporting the star schema.

 **Note:**

Do not use the same Financial Management database from which you are exporting data as the destination database.

9. For **Relational Table Prefix**, enter a prefix that to identify the tables for the star schema, or use the default application name prefix.

 **Note:**

The prefix can contain up to 10 alphanumeric characters and must start with a letter. It cannot contain an underscore. The default prefix is the application name.

10. Click **Extract**.

## Deleting a Star Schema

You can delete a star schema that you no longer need. Deleting a star schema deletes all of the star schema data, metadata, and tables.

To delete a star schema:

1. Select **Consolidation**, then **Extract**, and then **Data**.
2. From **Extract Destination Type**, select **Database**.
3. From **Destination Database (DSN)**, select the database that contains the star schema.
4. For **Relational Table Prefix**, enter the prefix that identifies the star schema's tables.

 **Note:**

The prefix can contain up to 10 alphanumeric characters and must start with a letter. The default prefix is the application name.

5. From **Schema Actions**, select **Delete Star Schema**.

## Creating a Star Schema Template

You can create a star schema template, which enables you to name and save POVs so that you can use them again.

To create a star schema template:

1. Select **Consolidation**, then **Extract**, and then **Data**.
2. Select the POV dimension members.
3. From **Destination Database (DSN)**, select the database to which you are exporting the star schema.
4. For **Relational Table Prefix**, type the prefix that identifies the star schema's tables.

 **Note:**

The prefix can contain up to 10 alphanumeric characters and must start with a letter. The default prefix is the application name.

5. Click **Save Template**.
6. Enter a template name and click **OK**.

## Deleting a Star Schema Template

To delete a star schema template:

1. Select **Consolidation**, then **Extract**, and then **Data**.
2. From **Template**, select the template to delete.
3. Click **Delete Template**.
4. At the system prompt, click **OK**.

# 9

## Defining Reports

You can define these report types in Oracle Hyperion Financial Management:

- Journal reports, which display information for a specific journal or list of journals based on criteria that you select in the Journals module
- Intercompany Partner (ICP) Matching reports, which display the intercompany transactions that are to be eliminated during consolidation

You can create a new report definition for each report, or you can open and modify a report definition.

You define a report using one of these methods:

- Specifying values using the Report page options. See the *Oracle Hyperion Financial Management User's Guide*.
- Using a text editor to specify report options in scripts

The date, time, and user fields are automatically displayed as header information on all reports. The Point of View definitions differ for each report. For detailed examples of reports, see [Defining Journal Report Scripts](#), and [Defining Intercompany Matching Report Scripts](#).

Sample Intercompany Partner Matching report scripts are included when you install Sample Applications for Financial Management. The files are located in the Sample Applications folder in the directory to which you installed Financial Management.

### Defining Journal Report Scripts

You create journal reports to check the status of journals and review journal adjustments. You can create a journal report to display information for a specific journal, or list of journals. For example, you can select to display only journals with a specific status, balance type, entity, or account. You can also select the columns that display on the report and change their sort order.

You can set entity and account filters for journal reports in the report definition. The syntax for Entity filtering is `EntityFilter=ParentName.ChildName` (for example, `UnitedStates.Maryland`). The syntax for Account filtering is `AccountFilter=MemberName` (for example, `Sales`).

This example shows a sample Journal report definition:

```
ReportType=Journal

ReportDescription=Tax Journals

POV=S#Actual.Y#2014.P#January.V#<Entity Curr Adjs>

DisplayColumn_0=Account,Ascending,NoRepeat,Yes,Label

DisplayColumn_1=ICP,,NoRepeat,No,Label

DisplayColumn_2=Products,,NoRepeat,No,Label

DisplayColumn_3=Markets,,NoRepeat,No,Label

StatusFilter=Working,Submitted,Rejected,Approved,Posted

TypeFilter=Regular

BalanceTypeFilter=Balanced,Unbalanced

EntityFilter=UnitedStates.Maryland

AccountFilter=Sales
```

## Defining Intercompany Matching Report Scripts

Intercompany Matching reports help you track intercompany transactions for analysis and auditing purposes. The Intercompany Matching report shows matches for entities and intercompany partner dimensions that have been set up with the Intercompany Partner (IsICP) attribute enabled.

Intercompany Partner (ICP) Matching reports list the intercompany transactions that are eliminated during consolidation. Intercompany transactions are managed across the Intercompany Partner dimension. The Intercompany dimension contains all intercompany balances that might exist for an account. You can enter intercompany transactions through data grids, data loads, journals, or data forms. Oracle Hyperion Financial Management can track and eliminate intercompany transaction details across accounts and custom dimensions. Intercompany transactions are eliminated as of the first common parent of two intercompany entities. They are eliminated through the [Elimination] member of the Value dimension.

You can create Intercompany Matching reports by using the user interface

You can select accounts for the report, or use the plug account option, in which the system generates the account and matching account based on the plug account. You can select to suppress reversed transactions, custom dimensions or intercompany details. When you print a report, you can override report settings to customize the report for your needs.

## Selecting Member Lists for Intercompany Matching Reports

You specify dimension members that comprise the Point of View for the generated report. You can select member lists for the report Entity and Partner. For example, you could select the Regional member list for the Entity field. If you select a member list for Entity and Partner, the system processes the ICP transactions for all entities in the Entity list against all entities in the Partner list. The system only processes transactions for entities that have the ICP attribute enabled.

## Selecting Accounts for Intercompany Matching Reports

You must define the account and matching accounts for which you want to match transactions. If you want the system to generate the account and matching account based on the plug account, you can specify the plug account option and the system automatically generates the account and matching account.

## Specifying Decimal Places in Intercompany Matching Reports

You can add the Decimal keyword to an Intercompany Matching report definition to specify the number of decimal places to display in the report. The possible values for the decimal are default, 0-9. If you do not specify a decimal value, the system uses the default decimal setting as defined in the account.

## Selecting Style Sheets for Intercompany Matching Reports

You can select a style sheet from a drop-down list when you are creating a report. You can also add the StyleSheet keyword to the report definition to specify the style sheet to use for the report; for example, `StyleSheet=HFM_IntercompanyDefault.xsl`. If you do not specify a style sheet in the report definition, the system uses the default style sheet.

## Specifying Currencies in Intercompany Matching Reports

You can generate an Intercompany Matching report in a currency that you specify. This enables you to run the report and validate intercompany transactions in a common currency before the amounts are consolidated. For example, to check the values in the EUR currency, you could change the Value in the Point of View from USD to EUR and generate the report.

If you define a report using a currency that has not been translated, the system performs the translation process using the translation rules defined in the Sub Translate section of your rule file. The system also stores the translated amounts in the corresponding currency Value dimension member. However, if the reporting currency has previously been translated and the translation status of the entity is OK, the system does not need to re-translate and uses the stored translated amounts for processing the Intercompany Matching report.

For example, if you run an Intercompany Match report for the currency EUR, the system first checks if the translated data has been created for the EUR currency (V#EUR). The system also checks to ensure that translation status is OK. If the translation status of the entity is TR (requires translation), the system re-translates to ensure that the translated data is valid. Otherwise, the system uses the stored translated amounts for processing. However, if the data



has not yet been translated to EUR, the system performs the translation process as defined in Sub Translate and stores the translated amounts in the EUR value member. The EUR translated amounts are also used for the Intercompany Matching report.

## Suppression Options for Intercompany Matching Reports

When you create an Intercompany Matching report, you can suppress several types of intercompany transactions and detail from the report. You can also select these suppression options when you print the report.

### Suppressing Matches

When you create an Intercompany Matching report, you can show or suppress matching Entity/Partner transactions. If you select to suppress them, the system suppresses the transactions if the Entity and Partner amounts are within the matching tolerance amount or percentage. For example, if an Entity has an amount of 299 and the Partner has 200, the difference is 99. If the matching tolerance amount is 100 and the difference between the Entity and Partner is less than 100 as in this example, the system suppresses the transactions because it is within the matching tolerance.

If you select not to suppress the Entity/Partner transactions, the system does not suppress them even if the Entity and Partner amounts are within the matching tolerance amount or percentage. Using the previous example, even if the difference amount is 99 and if it is within the matching tolerance, the system does not suppress the transactions. The matching tolerance specified is displayed in units.

### Suppressing Reversed Transactions

By default, the system displays the reversed transactions from an Intercompany Partner every time a transaction is displayed for the Entity. You can select to suppress these reversed transactions when you generate a report.

This option is very useful when you have one matching account for the report. For example, you might have a Cash account used to store intercompany transactions for Revenue and Expense. In this case, you must use the Suppress Reversed Transaction option to avoid a double entry because there is only one matching account for the report.

### Suppressing Details

If you want to display only the different amount in the report, you can suppress the intercompany details. When you select this option, the report does not display the intercompany transactions and prints only the total difference for each Entity/Partner section. If there is a discrepancy and you need to view each intercompany transaction, you can regenerate the report and show intercompany details.

### Suppressing Custom Dimensions

You can select to suppress the columns for any of the Custom dimensions.

### Member Display Option

You can display the label, description, or both for the dimension member in the report.

### Group By Option

You can group your intercompany partner transactions by Custom dimension. The system sorts the details based on this option and provides a subtotal for the group.

## Intercompany Matching Report Script Keywords

You use the keywords in this section to define Intercompany Partner Matching report scripts. After you create a script, save it with the RPT file name extension.



### Note:

Intercompany Partner Matching report script keywords are not case-sensitive.

### ReportType

This keyword specifies the report type. This keyword is required in the script.

#### Syntax

```
ReportType=Intercompany
```

### ReportDescription

This keyword specifies the description for the report. The report description can contain a maximum of 40 characters. This keyword is required in the script.

#### Syntax

```
ReportDescription=ReportDescription
```

Replace *ReportDescription* with the description for the report. For example:

```
ReportDescription=Intercompany Elimination Report
```

### StyleSheet

This keyword specifies the style sheet to use for the report.

#### Syntax

```
StyleSheet=StyleSheetFileName
```

Replace *StyleSheetFileName* with the style sheet for the report. For example:

```
ReportDescription=HFM_IntercompanyDefault.xsl
```

If you do not specify a style sheet in the report definition, the system uses the default style sheet.

### POV

This keyword specifies the point of view for the report. This keyword is required in the script.

## Syntax

```
POV=S#Scenario.Y#Year.P#Period.V#Value.W#View
```

Replace *Scenario*, *Year*, *Period*, *Value*, and *View* with valid dimension members. For example:

```
POV=S#Actual.Y#2014.P#January.V#USD.W#YTD
```



### Note:

Adjs value members, for example parent curr adjs, are not supported.

## Entity

This keyword specifies the entity or entity member list to be displayed on the report.

### Syntax

```
Entity=E#Parent.Entity
```

```
Entity=E{EntityList}
```

Replace *Entity.Parent* with the entity-parent combination. Replace *EntityList* with the name of a valid member list. For example:

```
Entity=E#UnitedStates.Connecticut
```

```
Entity=E{Geographical.[Base]}
```

## Partner

This keyword specifies the partner or partner member list to be displayed on the report.

### Syntax

```
Partner=E#PartnerParent.PartnerEntity
```

```
Partner=E{PartnerList}
```

Replace *PartnerParent.PartnerEntity* with the partner parent-entity combination. Replace *PartnerList* with the name of a valid partner member list. For example:

```
Partner=E#UnitedStates.Florida
```

```
Partner=E{Geographical.[Base]}
```

## AccountEntity and AccountPartner

The *AccountEntity\_x* and *AccountPartner\_x* keywords specify the accounts for matching. For each account pair to match, you specify *AccountEntity\_x* and *AccountPartner\_x* starting with zero. To create a One-to-Many or Many-to-Many matching report, specify additional accounts using the correct keyword. You cannot use duplicate keywords within one report. For example, *AccountEntity\_0* cannot exist more than once within one report.

### Syntax

```
AccountEntity_0=A#Sales.C1#GolfBalls.C2#Customer2
```

```
AccountPartner_0=A#Purchases.C1#GolfBalls.C2#Customer2
```

This example uses these keywords to show all accounts in one matching report:

AccountEntity\_0=A#1004780

AccountEntity\_1=A#1004790

AccountEntity\_2=A#1005850

AccountEntity\_3=A#1005850

AccountPartner\_0=A#2000100

AccountPartner\_1=A#2000140

AccountPartner\_2=A#2000210

AccountPartner\_3=A#2000250

AccountPartner\_4=A#2000320

AccountPartner\_5=A#2000430

AccountPartner\_6=A#2000560

AccountPartner\_7=A#2000630

AccountPartner\_8=A#2000680

This example uses the Entity and Partner keywords to create different reports with different account pairs. This is an example of one report with one account matching two accounts (one to many):

Report 1

AccountEntity\_0=A#1004780

AccountPartner\_0=A#2000100

AccountPartner\_1=A#2000140

This example shows one report with two accounts matching one account (many to one):

Report 2

AccountEntity\_0=A#1004790

AccountEntity\_1=A#2000210

AccountPartner\_1=A#2000250

Report 3 is an example of one report with one account matching with one account (one to one).

Report 3

AccountEntity\_0=A#1005850

AccountPartner\_0=A#2000320

Report 4 is an example of one report with two accounts matching with four accounts (many to many).

Report 4

AccountEntity\_0=A#1005850

AccountEntity\_1=A#1005860

AccountPartner\_0=A#2000430

AccountPartner\_1=A#2000560

AccountPartner\_2=A#2000630

AccountPartner\_3=A#2000680

## SuppressIfMatch

This keyword suppresses transactions if the entity and partner amounts are within the matching tolerance amount.

## Syntax

```
SuppressIfMatch=Boolean
```

Replace *Boolean* with Yes or No. For example:

```
SuppressIfMatch=Yes
```

For example, if an Entity has an amount of 299 and the Partner has 200, the difference is 99. If the matching tolerance amount is 100 and the difference between the Entity and Partner is less than 100, as in this example, the system suppresses the transactions because it is within the matching tolerance.

If you select not to suppress the Entity/Partner transactions, the system does not suppress these even if the Entity and Partner amounts are within the matching tolerance amount. Using the previous example, even if the difference amount is 99 and if it is within the matching tolerance, the system does not suppress the transactions.



### Note:

Matching tolerance is specified in units.

## SuppressReversedTransactions

This keyword suppresses reversed transactions from partners for each corresponding entity transaction.

### Syntax

```
SuppressReversedTransactions=Boolean
```

Replace *Boolean* with Yes or No. For example:

```
SuppressReversedTransactions=Yes
```

By default, the system displays the reversed transactions from an Intercompany Partner every time a transaction is displayed for the Entity. You can select to suppress these reversed transactions when you create a report.

This option is useful when you have one matching account for the report, for example, if you have one "wash" account to store intercompany transactions for Revenue and Expense. In this case, you must use the suppress reversed transactions option to avoid a double entry because there is only one matching account for the report.

## SuppressDetails

This keyword suppresses intercompany detail and prints only the Total difference for each Entity/Partner section.

## Syntax

```
SuppressDetails=Boolean
```

Replace *Boolean* with Yes or No. For example:

```
SuppressDetails=Yes
```

To create an Intercompany Matching report that displays only the difference amount, you can suppress the intercompany details when you create it. When you select this option, the report does not display the intercompany transactions and only prints the Total difference for each Entity/Partner section. If there is a discrepancy and you need to view each intercompany transaction, you can regenerate the report and show intercompany details.

## MatchingTolerance

This keyword enables you to specify a value to view only out-of-balance transactions over a certain amount, or use the default value of 0.



### Note:

Matching tolerance is specified in units.

## Syntax

```
MatchingTolerance=ToleranceValue
```

Replace *ToleranceValue* with a number that is less than 1 billion. The limit for this keyword is 999999999. For example:

```
MatchingTolerance=100
```

For example, if an Entity has an amount of 299 and the Partner has 200, the difference is 99. If the matching tolerance amount is 100 and the difference between the Entity and Partner is less than 100 as in this example, you can use the SuppressIfMatch keyword to have the system suppress the transaction because it is within the matching tolerance.

## SuppressCustoms

This keyword suppresses Custom dimensions.

## Syntax

```
SuppressCustomAlias=Boolean
```

Replace *Boolean* with Yes or No. The default is Yes. For example:

```
SuppressCustomFlows=Yes
```



## ScaleFactor

This keyword specifies the unit in which amounts are displayed by identifying where the decimal point is placed. For example, if you enter a scale factor of 3, the report amount is displayed in thousands. If the scale factor is 0, the report amount is displayed in units.

### Syntax

```
ScaleFactor=Scale
```

Replace *Scale* with one of these numbers:

- 0 = Units
- 1 = Tens
- 2 = Hundreds
- 3 = Thousands
- 4 = Ten Thousands
- 5 = Hundred Thousands
- 6 = Millions
- 7 = Ten Millions
- 8 = Hundred Millions
- 9 = Billions

For example:

```
ScaleFactor=3
```

In this example, the number 12345.78 is displayed as 12.345678 on the report.

## Decimal

This keyword specifies the number of decimal places to display in the report and can override the number of decimal places defined in the account.

### Syntax

```
Decimal=NumberDecimalPlaces
```

Replace *NumberDecimalPlaces* with a number 0-6. If you do not specify a decimal value, the system uses the default decimal setting as defined in the account. For example:

```
Decimal=3
```

In this example, the number 123.4567 is displayed as 123.457.

## DisplayLabels

This keyword specifies if member labels are displayed on the report.

### Syntax

```
DisplayLabels=Boolean
```

Replace *Boolean* with Yes or No. The default is Yes. For example:

```
DisplayLabels=Yes
```

## DisplayDescriptions

This keyword specifies if member descriptions are displayed on the report.

### Syntax

```
DisplayDescriptions=Boolean
```

Replace *Boolean* with Yes or No. The default is No. For example:

```
DisplayDescriptions=Yes
```

## DisplayPlugElimAccts

This keyword specifies if a summary of plug accounts affected by the intercompany transactions is displayed.

### Syntax

```
DisplayPlugElimAccts=Boolean
```

Replace *Boolean* with Yes or No. For example:

```
DisplayPlugElimAccts=Yes
```

## GroupByCustom

This keyword groups Custom dimension transactions by dimension. This keyword is optional.

### Syntax

```
GroupByCustom=Custom
```

Replace *Custom* with the Custom dimension alias by which to group the Custom dimensions in the report. For example:

```
GroupByCustom=Flows
```

## Defining Intercompany Transaction Report Scripts

You can define these types of intercompany transaction reports:

- IC Transactions - create a list of transactions.
- Intercompany Match By Account- create matching reports based on accounts selected.
- Intercompany Match By ID- create matching reports based on transaction ID.

**Table 9-1 Keywords for Intercompany Transaction Report Scripts**

Keyword	Description
ReportType	Specify the report type. For example, ReportType=ICTransactions
ReportLabel	Specify the report name. For example, ReportLabel=Transaction Report
ReportDescription	Specify a report description. For example, ReportDescription=Intercompany Transaction Detail Report
ReportSecurityClass	Specify the security class for the report. The default is [Default].
POV	Specify a valid point of view for the report. For example, POV=S#ActMon.Y#2014.P#January
ScaleFactor	Specify a scale factor for the report. The scale can be a value from 0 to 9.
Decimal	Specify the number of decimals to display in the report. The number of decimals can be a value from 0 to 9.
FromAmt and ToAmt	Specify a range of transaction amounts.
IncludeMatched	Specify True to include matched transactions in the report, otherwise False.
IncludeUnMatched	Specify True to include unmatched transactions in the report, otherwise False.
IncludeMisMatched	Specify True to include mismatched transactions in the report, otherwise False.
IncludePosted	Specify True to include posted transactions in the report, otherwise False.
IncludeUnPosted	Specify True to include unposted transactions in the report, otherwise False.
Entity	The entity must be a valid ICP base entity, not a parent entity.
Partner	The partner entity must be a valid ICP entity for the account.
Entity Account	If you are displaying Entity transactions, specify an entity account.
Partner Account	If you are displaying Partner transactions, specify a partner account.

**Table 9-1 (Cont.) Keywords for Intercompany Transaction Report Scripts**

Keyword	Description
TransactionID	This ID is required. You must enter an ID for the transaction, with a maximum of 40 characters. When combined with the Sub ID, this ID becomes a unique identifier for the Entity/Partner/Account/C1/C2 within the Scenario/Year/Period.
TransactionSubID	Specify a transaction Sub ID.
TransactionCurrency	This is the currency used for the Invoice transaction. It must be a valid currency defined in the application.
ReferenceID	This is optional. You can enter a Reference ID to store reference information for the transaction. For example, the entity might have its own set of invoice numbering that is different from the entity that issued the invoice. You can enter additional information in this Reference ID for information purposes only. You can enter the entity reference ID in the Transaction ID and enter the corresponding invoice number from the partner entity.
MatchCode	This is optional. The match code must be one of these prefixes to distinguish the different types of matching processes: <ul style="list-style-type: none"> <li>• A - Auto-matching performed using Accounts</li> <li>• I - Auto-matching performed using the Transaction ID</li> <li>• R - Auto-matching performed using the Reference ID</li> <li>• M - Manual matching performed</li> </ul>
ReasonCode	This is optional. The reason code must be a valid reason code defined by the administrator. The main purpose of the reason code is to indicate why a transaction has a MisMatched status - for example, because of a missing invoice from the partner entity, or an incorrect amount entered by the partner. If the transaction has a Matched status, you do not need to assign a reason code for the transaction. You cannot assign a reason code to transactions with an UnMatched status.
FromDate	Optional. This must be a valid date.
ToDate	Optional. This must be a valid date.
DisplayColumns Section	Specifies the columns that are displayed in the report and how they are displayed. Syntax is: <DisplayColumns>, <Sort>,<Label>, <Repeat>, <Totals> where <Displayed Columns> is a valid column, <Sort> is Sort or No Sort, <Label> is a label, description, or both, <Repeat> is Repeat or NoRepeat, and <Total> is Total or NoTotal. For Custom dimensions, the alias is used for the column, for example: Customers,NoSort,Label,Repeat,NoTotal

**Table 9-1 (Cont.) Keywords for Intercompany Transaction Report Scripts**

Keyword	Description
DisplayEntityTransactions	<p>Specify True to display entity transactions in the report, otherwise False.</p> <p>You can select to display only intercompany transactions for a specific entity and partner, or also display the corresponding transactions from the partner with the entity. For example, if you select only the transactions for Entity A with Partner B in the Entity and Partner selections, the system displays only the transactions that Entity A has with Partner B. However, if you want to see the corresponding transactions for Entity B with Partner A, you can select to include Entity transactions and Partner transactions.</p>
DisplayPartnerTransactions	<p>Specify True to display partner transactions in the report, otherwise False.</p>
SuppressDetails	<p>Specify True to suppress transaction detail and display only the subtotal row.</p>

# 10

## Managing Rules

You use Oracle Hyperion Financial Management rules to automate the calculations of data within an application. You can use rules for these purposes:

- Calculate data entry level amounts for a specific entity, scenario, and period.
- Prevent data entry for a specific cell in a specific entity, scenario, and period.
- Allow input at the Parent entity level.
- Calculate data that cannot be calculated through a hierarchical aggregation, such as ratios or variance analysis.
- Perform allocations from a parent entity to a list of base entities.
- Perform complex currency conversions, calculate exchange rate differences, or perform other calculations necessary for your consolidation.
- Define formulas to dynamically calculate accounts.
- Specify the accounts in the application that support intercompany transactions.

You can write rules in a text editor, such as Notepad ++.



### Note:

To work with rules in Oracle Hyperion Calculation Manager, see [Creating Rules Using Calculation Manager](#).

## Rule Types

You can write rules that set and clear values, calculate data, translate currency, consolidate data, allocate data from one location to another, and prevent data input.

**Table 10-1 Rule Types**

Rule Type	Description
Calculation	Calculation rules run when users run calculations. You can use Calculation rules to perform calculations that cannot be calculated through the natural order of the dimension hierarchies. For example, you can create calculations to derive ratios or opening balances. The Calculate() routine is executed when you calculate or consolidate data.

Table 10-1 (Cont.) Rule Types

Rule Type	Description
Translation	<p>Translation rules run when users run translations. You can use Translation rules to perform calculations related to non-standard translations. The Translate() routine is executed when you translate or consolidate data.</p> <p>For example, if the application is using the default ClosingRate to translate Assets and LIABILITY accounts and AverageRate to translate REVENUE and EXPENSE accounts, you may want to use a different translation rate to calculate the translation difference related to Net REVENUE.</p> <p>Oracle Hyperion Financial Management executes Translation rules in these cases:</p> <ul style="list-style-type: none"> <li>• When a user runs a currency translation by right-clicking in a data grid and selecting Translate or Force Translate from the menu that is displayed. This is useful for performing translations as you enter data.</li> <li>• When a user performs a consolidation and a parent entity's default currency is different from a child entity's default currency.</li> </ul>
Consolidation	<p>Consolidation rules run when users run consolidations. You can use consolidation rules to perform non-standard consolidations, most commonly found in statutory applications. The Consolidate() routine is executed when you consolidate data.</p>
Allocation	<p>Allocation rules allocate data from one entity to a list of entities. For example, you can use Allocation rules to calculate administrative expenses for a list of entities based on the total administrative expenses for another entity.</p>
Input	<p>Input rules allow input at the Parent entity level. Only the entity currency Value dimension is supported. Note that contribution values from children to the Parent entity's entity currency Value dimension are not rolled up. The contribution value for the Parent.Child combination is stored.</p>

Table 10-1 (Cont.) Rule Types

Rule Type	Description
NoInput	<p>NoInput rules prevent input at the Base entity level, thus reserving the cells for calculations. You can use the NoInput function multiple times in a NoInput rule to prohibit data input into several non-adjacent cells.</p> <p>These limitations and guidelines apply to NoInput rules:</p> <ul style="list-style-type: none"> <li>• Only these Financial Management functions are supported for NoInput rules: <ul style="list-style-type: none"> <li>– NoInput</li> <li>– List</li> </ul> <p>All VBScript methods are supported for NoInput rules.</p> </li> <li>• For the List function, fixed lists, system lists, and dynamic lists are supported. A dynamic list can reference metadata attributes supported by the List function.</li> <li>• An If...Then structure can test for metadata attributes. However, NoInput rules do not support testing of members in the current Point of View. For example, you cannot test <code>If HS.Entity.Member = "CT"</code>.</li> <li>• Be careful when using the NoInput function in loops. A few simple statements with loops may end up loading thousands of cells in memory, so be sure to test the performance impact of loops that include NoInput.</li> </ul>
Dynamic Calculation	<p>Dynamic rules enable you to define formulas to dynamically calculate accounts. You can dynamically calculate Base accounts only. You cannot use Dynamic rules on Parent accounts.</p> <p>Use these guidelines for writing dynamic calculation rules:</p> <ul style="list-style-type: none"> <li>• The right side of the equation must reference the same Scenario/Year/Entity combination. This means you cannot reference prior year amounts in your calculations.</li> <li>• Only dynamic accounts are valid on the left side of the equation.</li> <li>• Dynamic accounts cannot be used on the right side of the equation.</li> <li>• Only Account and View are valid on the left side of the equation.</li> <li>• If View is not specified, the calculation executes for YTD and Periodic. If View is specified, the calculation is executed only for the specified view.</li> <li>• HS.View.PeriodNumber is the only HS statement that can be used in a HS.Dynamic calculation.</li> <li>• All statements in the Sub Dynamic section are executed sequentially.</li> </ul>



**Table 10-1 (Cont.) Rule Types**

Rule Type	Description
Transactions	Transactions rules specify the accounts in the application that support intercompany transactions. Cells supporting transactions are read-only in data grids and forms.
Equity Pickup	Equity pickup rules specify the owned entity, owner entity, and percentage of ownership This is the default point of view when the Sub EquityPickup section is run: <ul style="list-style-type: none"> <li>• Current scenario, year, and period</li> <li>• Entity: owner of the pair processed</li> <li>• Value: &lt;Entity Currency&gt;</li> </ul>
OnDemand	On-demand rules are used in Data Forms, and enable you to run a subset of calculations to quickly see the results in the data form. All HS functions that can be used in Sub Calculate (but no others) can also be used in OnDemand rules.

You create rules for Financial Management in a unique script, which is based on the Microsoft VBScript language. Rules are constructed through the combination of functions, objects, and other arguments to generate the scripting syntax. Within each routine, you use two types of functions to write rules:

- Financial Management functions that define calculations
- VBScript functions that are used to write conditional statements

## Rule Considerations

Following are considerations for writing rules for applications:

- Rules are executed when users perform calculations, translations, consolidations, and allocations. Calculation rules execute one time for each Entity/Value dimension intersection to which the calculation or consolidation applies. See [Calculation Rules with Calculation Commands](#).
- The dimension members to which Oracle Hyperion Financial Management applies calculations depend on the data grid cell where the user's cursor is placed and the members specified in the Point of View bar. See [Current Dimension Members](#).
- When a function puts data into a currency-related Value member, Financial Management might delete the current value in the Value member. See [Functions Automatically Clear Data](#).
- Rules execute in sequential order within a routine and there is a set order in which routines run before other routines. See [Rule Execution During Consolidation](#).

## Calculation Commands

Calculate routines are executed for a specified intersection of scenario, year, period, entity, and value. Executing Calculate routines results in writing or clearing data in the current data table, which corresponds to the current intersection of scenario, year, period, entity, and value. When

users execute Calculate routines, the system can read data from anywhere in the application. However, data is only written to the current data table.

 **Note:**

The calculate routine will fail if the [None] entity exists in a hierarchy when you consolidate.

You run these processes in data grids by selecting one of these commands:

- Calculate
- Force Calculate
- Calculate Contribution
- Force Calculate Contribution
- Consolidate
- Consolidate All With Data
- Consolidate All
- Translate
- Force Translate

When you select a calculation command, Oracle Hyperion Financial Management executes the Sub Calculate() routine in the RLE file. The Calculate() routine calculates accounts and custom dimension members for a specified Entity-Value combination within a specified Scenario, Year, and Period.

## Force Calculate

The Force Calculate option forces rules to run only on the Value member selected, and any Value member on which it depends. For example, Force Calculate on entity currency runs rules on the entity currency member only. Force Calculate on the entity currency total runs rules on entity currency, entity currency adj, and entity currency total. Force Calculate is the only command for which it is possible to affect only a single member of a value triplet.

## Calculation Rules with Calculation Commands

The number of times that a Calculation rule is executed depends upon the calculate command selected by the user and by other factors.

 **Caution:**

Carefully read these sections before writing rules. You might want an operation to occur only for certain members of the Value dimension, and if this is the case you must test for the current member with VBScript's If structure and the `Member` function before executing the operation. For steps and examples on using If structures and the `Member` function, see [Conditional Rules](#).

When a user selects the Calculate or Force Calculate command, Oracle Hyperion Financial Management runs the application's Calculation rule for the intersection of the current entity member and the Value member for the entity's default currency, Entity Currency. If the entity's AllowAdjs attribute is enabled in the metadata, Financial Management also runs the rule a second time, applying the rule to the intersection of the entity and the member of the Value dimension that stores adjustments to the entity's default currency, Entity Curr Adjs.

### Example

For example, if an entity named California has a default currency of USD and its AllowAdjs attribute is enabled in the metadata, the calculation rule runs twice, once for the intersection of California with USD and once for the intersection of California with USD Adjs.

## Calculation Rules with Consolidation Commands

When a user selects one of the Consolidate commands, Oracle Hyperion Financial Management runs the Calculation rule for several of the Value dimensions that intersect each previously unconsolidated child entity. After executing for the children, Financial Management runs the rule for the intersection of the parent entity and the member of the Value dimension for the parent's default currency.

1. The rule is run for the intersection of the child entity and the Value member that stores the child entity's default currency (Entity Currency).
2. If the child entity's AllowAdjs attribute is set to Y, the rule is run for the intersection of the child entity and the Value member that stores adjustments in the child entity's default currency (Entity Curr Adjs).
3. If the child's currency differs from the parent's currency, the rule is run for the intersection of the child entity and the Value member that stores amounts translated to the parent's home currency (Parent Currency).
4. If a child's currency differs from the parent's currency and the rules file contains a Translation rule, Financial Management executes the Translation rule before step 3.
5. If the child's currency differs from the parent's currency, and the child entity's AllowAdjs attribute is set to Y, the rule is run for the intersection of the child entity and the Value member that stores adjustments translated to the parent's home currency (Parent Curr Adjs).
6. If the parent entity's AllowAdjFromChildren attribute is set to Y, the Logic rule is run for the intersection of the child entity and the Parent Adjs value.
7. The rule is run for the intersection of the child entity and the Proportion value.
8. The rule is run for the intersection of the child entity and the Elimination value.
9. If the parent entity's AllowAdjFromChildren attribute is set to Y, the rule is run for the intersection of the child entity and the Contribution Adjs value.
10. For each additional child entity that contains previously unconsolidated data, repeat steps 1 through 9.
11. The rule is run for the intersection of the parent entity and the Value member that stores the parent entity's default currency.

### Example

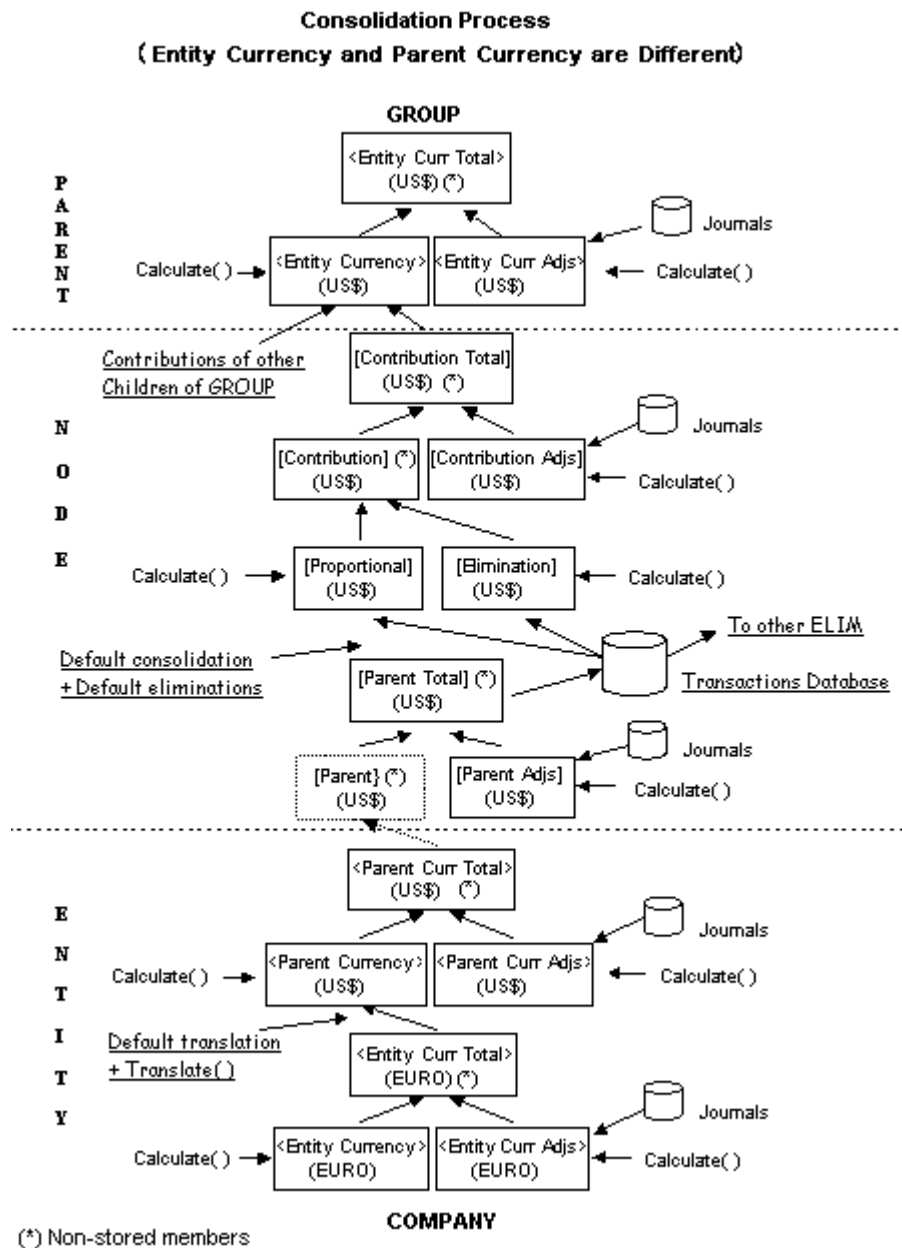
For example, a parent entity named UnitedStates has children named EastUS and WestUS. The children have the AllowAdjs attribute enabled. The UnitedStates entity has the AllowAdjs and the AllowAdjFromChildren attributes enabled. All three entities share a default currency of USD.

If you change data for EastUS and WestUS and consolidate UnitedStates, Financial Management runs the rule for each of these intersections of the Entity and Value dimensions:

1. EastUS and Entity Currency.
2. EastUS and Entity Currency Adjs. (EastUS's AllowAdjs attribute has been set to Y.)
3. EastUS and Parent.
4. EastUS and Parent Adjs. (UnitedStates' AllowAdjFromChildren attribute has been set to Y.)
5. EastUS and Proportion.
6. EastUS and Elimination.
7. EastUS and Contribution Adjs. (UnitedStates' AllowAdjFromChildren attribute has been set to Y.)
8. WestUS and Entity Currency.
9. WestUS and Entity Currency Adjs. (WestUS's AllowAdjs attribute has been set to Y.)
10. WestUS and Parent.
11. WestUS and Parent Adjs. (UnitedStates' AllowAdjFromChildren attribute has been set to Y.)
12. WestUS and Proportion.
13. WestUS and Elimination.
14. WestUS and Contribution Adjs. (UnitedStates' AllowAdjFromChildren attribute has been set to Y.)
15. UnitedStates and Entity Currency.

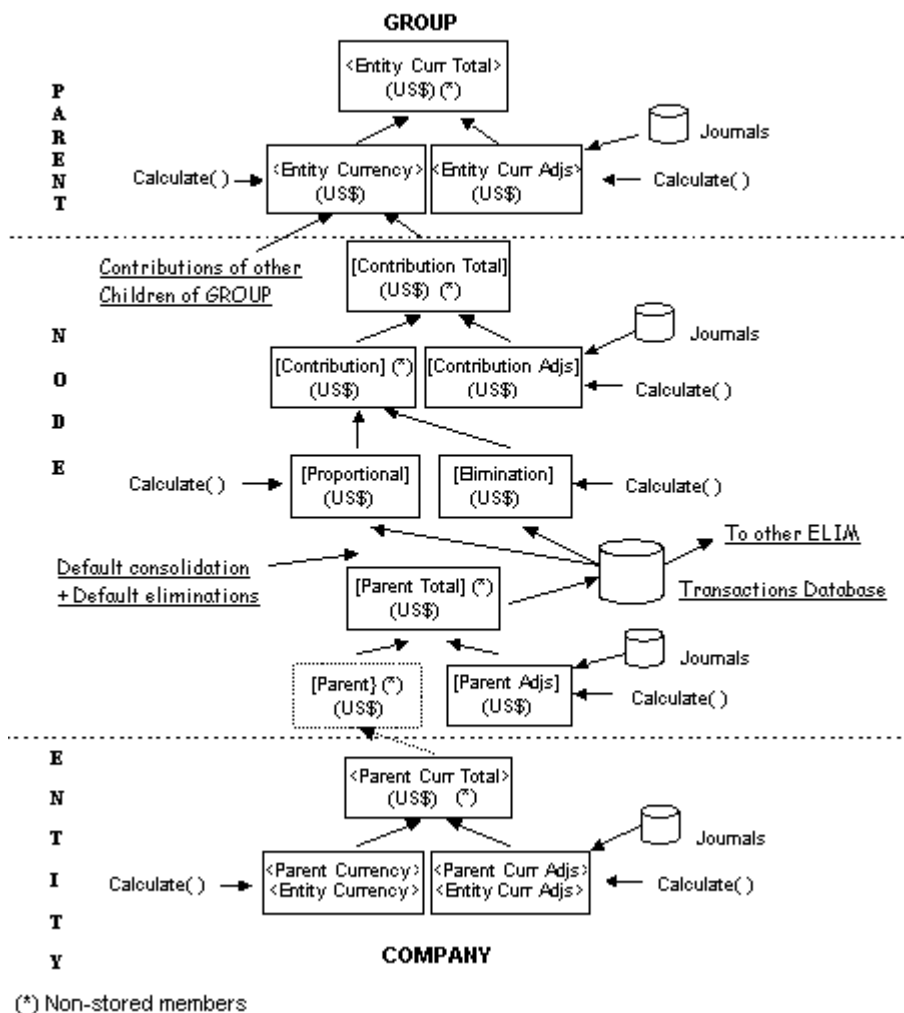
Following are examples of the consolidation process.

The first example shows the process when the entity currency and the parent currency are different.



The following example shows the process when the entity currency and the parent currency are the same.

**Consolidation Process  
(Entity Currency and Parent Currency are the Same)**



## Current Dimension Members

By default, Oracle Hyperion Financial Management applies Calculation rules to the current dimension members at the time that the user selects a calculation command. See [Calculation Rules with Calculation Commands](#). The current dimension members are determined by these factors:

- The cell in the data grid in which the user's cursor is placed when the user runs a calculation, translation, or consolidation.
- The dimension members specified in the Point of View bar.
- For consolidations, the current members of the Entity and Value dimensions change each time the rule is executed. See [Calculation Rules with Consolidation Commands](#).
- Rules process for the currently selected dimension members, except for calculation rules in which case the calculation rules process across all accounts in the application.

If a dimension member of the cell in which the cursor is placed differs from the corresponding member of the corresponding dimension in the Point of View bar, the cell's dimension member

overrides the Point of View bar's dimension member. For example, if a user has specified an entity named Europe in the Point of View bar and runs a translation with the cursor placed in a cell for an entity named Germany, Germany is the current entity.

By specifying a dimension member as an argument, you can use some functions to work with a specific dimension member regardless of the current dimension. For example, you can use the NumBase function with the Entity object to get the number of base entities for the entity that you specify in the argument.

For more complex functions, you can create Account Expressions to specify the dimension members.

## Account Expressions

Some functions require an Account Expression as an argument. In its simplest form, an Account Expression is a string that specifies the account to which Oracle Hyperion Financial Management applies the function.

The Account Expression characters are listed in the following table.

**Table 10-2 Characters Used to Specify Dimensions in Account Expressions**

Character	Dimension
A#	Account
I#	Intercompany Partner
C#	Custom
S#	Scenario
Y#	Year
P#	Period
W#	View
E#	Entity
V#	Value

To understand Account Expressions, consider the Clear function, which removes values from the dimension members specified in the function's argument. In this example, the argument "A#Sales" is an Account Expression:

```
HS.Clear "A#Sales"
```

The A# characters represent the Account dimension, and the word Sales is the member name of the Account dimension to which Financial Management applies the Clear function. This Account Expression tells Financial Management to clear the data stored in the Sales account.

When you use an Account Expression, Financial Management applies the function to the intersection of the account that you specify in the Account Expression and the current members of these dimensions:

- Entity
- Period
- Scenario
- Value

- View
- Year
- Custom - Uses the CustomTopMember that was set for the account in the metadata. For example, if the Account Expression does not specify a member of the Custom 3 dimension, Financial Management uses all valid Custom 3 members as defined by the CustomTopMember specified for the account.
- Intercompany Partner - All valid ICP members.

You can override the Intercompany and Custom dimension defaults by specifying members in the Account Expression. Each dimension is represented by certain characters. When you include more than one dimension in an Account Expression, you must separate the dimensions with periods.

When you create an Account Expression, you do not have to specify all of these dimension members; you can specify the members to which to apply the function. For example, this line clears the intersection of the Sales account and the Hardware ICP:

```
HS.Clear "A#Sales.I#Hardware"
```

## Functions Automatically Clear Data

When a function puts data into a Value member that relates to currencies, Oracle Hyperion Financial Management automatically clears data from the member if either of these conditions apply:

- The intersecting Entity member is a parent.
- The intersecting Account member is a calculated account.

 **Tip:**

The currency-related Value members are the system-generated Entity Currency member and the user-defined currency members such as USD, EURO, LIRA.

In addition, if a function puts data into a Value member for adjustments, Financial Management automatically clears data from the member if the intersecting Account member is a calculated account.

 **Tip:**

The adjustment-related Value members are those that include the code "Adjs" in their name, for example, USD Adjs, EURO Adjs, LIRA Adjs.

## Error Messages

When Oracle Hyperion Financial Management detects a syntax error, it displays an error message that contains this information:

- The line number in the RLE file that is causing the error.
- The applicable Financial Management object and function.



- An error description.

## Rule Execution During Consolidation

During the consolidation process, rules are executed in a pre-defined sequence. For each base child of a specific parent, the calculation sequence for the various elements in the Value dimension takes place in this order:

1. Accounts defined as IsCalculated in the metadata are cleared in EntityCurrency.
2. Accounts defined as IsCalculated in the metadata are cleared in EntityCurrAdjs.
3. The Sub Calculate() routine is executed on EntityCurrency.
4. The Sub Calculate() routine is executed on EntityCurrAdjs.
5. The ParentCurrency data is cleared.
6. Default translation is applied to all accounts defined as Revenue, Expense, Asset, Liability for the total amount of EntityCurrency and EntityCurrAdjs. For accounts with the Flow or Balance attribute, translation is not applied by default, the total amount of EntityCurrency and EntityCurrAdjs is rolled up into Parent Currency.
7. The Sub Translate() routine is executed.
8. The Sub Calculate() routine is executed on ParentCurrency.
9. Accounts defined as "IsCalculated" in the metadata are cleared in ParentCurrAdjs.
10. The Sub Calculate() routine is executed on ParentCurrAdjs.
11. Accounts defined as "IsCalculated" in the metadata are cleared in ParentAdjs
12. The Sub Calculate() routine is executed on ParentAdjs.
13. Proportion and Elimination data are cleared.
14. Default consolidation and eliminations are performed for the total amount of Parent and ParentAdjs.
15. The Sub Calculate() routine is executed on Proportion and Elimination.
16. Accounts defined as "IsCalculated" in the metadata are cleared in ContributionAdjs.
17. The Sub Calculate() routine is executed on ContributionAdjs.

After the previous steps have been repeated for each base child, this sequence takes place for the parent entity:

1. The EntityCurrency data is cleared.
2. The sum of the total of Proportion, Elimination, and ContributionAdjs for every child is written into EntityCurrency of the parent entity.
3. The Sub Calculate() routine is executed on EntityCurrency.
4. Accounts defined as "IsCalculated" in the metadata are cleared in EntityCurrAdjs.
5. The Sub Calculate() routine is executed on EntityCurrAdjs.

 **Note:**

If a parent is further consolidated into another parent, this sequence continues with step 5 from the child consolidation sequence.

## Default Translation

Following is the sequence in which default translation takes place.

1. The system checks the current entity for the direct translation rate and uses that rate for translation.
2. If the translation rate is not found, the system derives the direct rate from the indirect rate in the current entity.
3. If neither the direct rate nor the indirect rate is found in the current entity, the system looks at the [None] entity and uses the direct rate.
4. If the direct rate is not found in the [None] entity, the system derives the direct rate from the indirect rate in the [None] entity.
5. If the indirect rate for the [None] entity does not exist, the system derives the rate by triangulation using the application currency in the [None] entity.
6. If triangulation fails, the entity is not translated.

 **Note:**

The system first looks for a translation rate within the current entity. If not found in the current entity, the system looks for a translation rate within the [None] entity. If the system cannot find a translation rate in the [None] entity, the system translates using triangulation. Triangulation is a way to convert balances from one currency to another using a third, common currency.

For example, if you want to convert Euro to Yen, but the system cannot find a direct or indirect rate to perform the translation, if Euro and Yen can both translate into USD then, using triangulation, the system can convert the Euro balance to USD and then convert the USD balance to Yen.

## Financial Management Objects

This section explains the syntax you must use to represent Oracle Hyperion Financial Management objects when using functions.

The top-level object in Financial Management is the HS object. This means that when you use a Financial Management function, the first three characters must be the letters HS followed by a period:

HS.

This example demonstrates how to write the Clear function:

```
HS.Clear
```

If you do not precede Clear with the HS. characters, an error occurs.

 **Caution:**

You cannot use the HS. characters before a standard VBScript function. If you do, an error occurs. Use the HS. characters only before a Financial Management function.

Some Financial Management functions apply only to objects that are children of the HS object. These functions require you to put the applicable object's name between the HS characters and the function name. These objects are children of the HS object and are used in Financial Management:

- Account
- AppSettings
- Custom
- DataUnit
- Entity
- ICP
- Node
- Parent
- Period
- Scenario
- Value
- Year

## Using VBScript in Rules

VBScript functions are used to write conditional statements in rules.

 **Note:**

Oracle Hyperion Financial Management does not support .NET objects exposed as COM scriptable interfaces. COM capable applications, such as VBScript, cannot access them through Rules or Member Lists.

## VBScript Operators

These VBScript operators are supported in Oracle Hyperion Financial Management rules:

- And
- Eqv
- Imp
- Is
- Mod

- Not
- Or
- Xor
- =
- <
- >
- <=
- >=
- <>
- +
- &
- -
- /
- \
- \*
- ^

## VBScript Statements

These VBScript statements are supported in Oracle Hyperion Financial Management rules:

- Call
- Const
- Dim
- Do...Loop
- Erase
- Exit
- For...Next
- ForEach...Next
- Function...End Function
- If...Then...Else If...Else...End If
- On Error Go to 0
- On Error Resume Next
- Option Explicit
- Private
- Public
- ReDim
- ReDim Preserve
- SelectCase...Case...End Select

- Set
- While...WEnd
- With...End With
- ' (single quote)
- Sub...EndSub

## VBScript Keywords

These VBScript keywords are supported in Oracle Hyperion Financial Management rules:

- Empty
- Nothing
- Null
- True
- False

## VBScript Functions

These VBScript functions are supported in Oracle Hyperion Financial Management rules:

### Date and Time Functions

Function	Description
Cdate	Converts a valid date and time expression to the variant of subtype Date
Date	Returns the current system date
DateDiff	Returns the number of intervals between two dates
DatePart	DatePart(interval,date[,firstdayofweek[,firstweekofyear]])
DateSerial	DateSerial(year,month,day)
Day	Returns a number that represents the day of the month (between 1 and 31, inclusive)
Hour	Returns a number that represents the hour of the day (between 0 and 23, inclusive)
IsDate	Returns a Boolean value that indicates if the evaluated expression can be converted to a date
Minute	Returns a number that represents the minute of the hour (between 0 and 59, inclusive)
Month	Returns a number that represents the month of the year (between 1 and 12, inclusive)
Month/Name	Returns the name of a specified month
Now	Returns the current system date and time
Second	Returns a number that represents the second of the minute (between 0 and 59, inclusive)
Time	Returns the current system time
Timer	Returns the number of seconds since 12:00 AM
Year	Returns a number that represents the year

## Format Functions

Function	Description
FormatNumber	Returns an expression formatted as a number
FormatFunctions	Returns an expression formatted as a date or time

## Conversion Functions

Function	Description
Asc	Returns the first letter in a string to ANSI code
CBool	Converts an expression to a variant of subtype Boolean
CByte	Converts an expression to a variant of subtype Byte
CDbl	Converts an expression to a variant of subtype Double
Chr	Converts the specified ANSI code to a character
Cint	Converts an expression to a variant of subtype Integer
CLng	Converts an expression to a variant of subtype Long
CStr	Converts an expression to a variant of subtype String

## Math Functions

Function	Description
Abs	Returns the absolute value of a specified number
Fix	Returns the integer part of a specified number
Int	Returns the integer part of a specified number
Rnd	Returns a random number less than 1 but greater than or equal to 0
Sgn	Returns an integer that indicates the sign of a specified number
Sqr	Returns the square root of a specified number

## Array Functions

Function	Description
Array	Returns a variant containing an array
Filter	Returns a zero-based array that contains a subset of a string array based on a filter criteria
IsArray	Returns a Boolean value that indicates whether a specified variable is an array
Join	Returns a string that consists of a number of substrings in an array

---

Function	Description
Lbound	Returns the smallest subscript for the indicated dimension of an array
Split	Returns a zero-based, one-dimensional array that contains a specified number of substrings
Ubound	Returns the largest subscript for the indicated dimension of an array

---

## String Functions

---

Function	Description
InStr	Returns the position of the first occurrence of one string within another. The search begins at the first character of the string.
InStrRev	Returns the position of the first occurrence of one string within another. The search begins at the last character of the string.
Left	Returns a specified number of characters from the left side of a string
Len	Returns the number of characters in a string
LTrim	Removes spaces on the left side of a string
RTrim	Removes spaces on the right side of a string
Trim	Removes spaces on both the left and the right side of a string
Mid	Returns a specified number of characters from a string
Replace	Replaces a specified part of a string with another string for a specified number of times
Right	Returns a specified number of characters from the right side of a string
Space	Returns a string that consists of a specified number of spaces
StrComp	Compares two strings and returns a value that represents the result of the comparison
StrReverse	Reverses a string
LCase	Converts a specified string to lowercase
UCase	Converts a specified string to uppercase

---

## Other Functions

---

Function	Description
CreateObject	Creates an object of a specified type
Eval	Evaluates an expression and returns the result
IsEmpty	Returns a Boolean value that indicates whether a specified variable has been initialized or not
IsNull	Returns a Boolean value that indicates whether a specified expression contains no valid data (Null)
IsNumeric	Returns a Boolean value that indicates whether a specified expression can be evaluated as a number

---

---

Function	Description
Round	Rounds a number

---

## VBScript Objects

These VBScript object are supported in Oracle Hyperion Financial Management rules:

### File System Objects

- CreateTextFile
- OpenTextFile
- DeleteFile
- FileExist
- FolderExist
- GetBaseName
- GetParentFolderName
- GetFile
- CopyFile

### File Objects

- OpenAsTextStream
- Size

### Text Stream Objects

- AtEndOfStream
- Close
- WriteLine
- ReadLine

### Err Objects

- Description
- HelpContext
- HelpFile
- Number
- Source
- Raise
- Clear

## Commonly Used Rules

These sections show you how to write some simple and commonly used rules. The descriptions of these rules contain step-by-step procedures for readers who are not



comfortable with VBScript. These procedures are followed by examples. If you have experience with VBScript, you might prefer to skip the procedures and instead focus on the examples.

 **Tip:**

These procedures assume that you have a rules file and code you write is placed in the Calculate() subroutine. See [Creating Rules Files](#).

## Reusing Data

Use the EXP function to insert data from one account into another account. EXP's argument contains the account to be set and the account from which the value is retrieved. The argument is in the form of an equation, with the target account on the left side of the equal sign and the source account on the right side.

 **Note:**

EXP inserts data into the intersection of an account with the current dimension members (see [Current Dimension Members](#)). In addition, you can use Account Expression characters to override the current Custom and ICP members.

In this example, the Calculation rule sets the PrevCash account to the value in the Cash account:

```
HS.EXP "A#PrevCash = A#Cash"
```

You can use Account Expression characters to specify dimension members on both sides of the equal sign in EXP's argument. See [Exp](#). This example inserts the data from the previous year's intersection of the PrevCash account and the Golf member of the Custom 3 dimension into the current year's intersection of PrevCash and Golf:

```
HS.EXP "A#PrevCash.C3#Golf = A#Cash.Y#Prior.C3#Golf"
```

 **Tip:**

The Prior keyword that follows the Y# Account Expression characters causes EXP to retrieve the previous year's data. There are several similar keywords that apply to Year and Period in Account Expressions. See [Period and Year Keywords](#).

## Setting Accounts by Calculating Amounts

Another common task is to calculate the amounts contained in two accounts and then insert the result into another account. The EXP function supports addition, subtraction, multiplication, and division on the right side of the equal sign in its argument.

In this example, the Calculation rule divides the Sales account's value by the UnitsSold account's value, and inserts the quotient in the AvgPrice account:

```
HS.EXP "A#AvgPrice = A#Sales / A#UnitsSold"
```

## Conditional Rules

You may want a rule to execute an action only when certain dimension members are the current members in the Point of View. For example, you might want an account's value to be calculated in one way when Actual is the current scenario and a different way when Budget is the current scenario.

### Tip:

For information on how Oracle Hyperion Financial Management determines the current dimension members, see [Current Dimension Members](#).

To do this, use the Member function in a VBScript `If` structure. Member gets the name of the current member of these dimensions:

- Entity (Use the Entity object to get the current entity or the Parent object to get the parent of the current entity.)
- Period
- Scenario
- Value
- Year

If structures enable you to execute statements only if certain conditions are true. The following sections show a few different ways of using `Member` with If structures to test for dimension members.

### Tip:

These sections cover only a few of the If structure aspects. For more details on If structures, consult Microsoft's VBScript documentation. (You can download VBScript documentation from Microsoft's Web site.)

## Testing for a Dimension Member

To have Oracle Hyperion Financial Management execute an action only if a particular dimension member is the current member, use an If structure that tests the return value of the Member function.

In this example, if the current scenario is Budget, Financial Management multiplies the amounts in the UnitsSold and Price accounts and inserts the product in the Sales account.

```
If HS.Scenario.Member = "Budget" Then

    HS.EXP "A#Sales = A#UnitsSold * A#Price"

End If
```

 **Tip:**

All If structures must begin with an `If...Then` statement and end with an `End If` statement. The actions to be executed if the condition is met are sandwiched between the `If...Then` and `End If` statements as shown above.

## Testing for More Than One Member

You can test for more than one member in an `If...Then` statement. In other words, you can execute an action for two or more members of a dimension. Consider the example in the Testing for a Dimension Member section. You might want the Sales account's value to be calculated if the current scenario is Budget or Forecast.

To test for more than one member, use two `Member` functions and VBScript's `Or` keyword in the `If...Then` statement. Place `Or` after the first `Member` function, then place the second `Member` function between `Or` and `Then`.

 **Tip:**

You can use this technique to test for more than two members. For each member to be tested, include an additional combination of the `Member` function and the `Or` keyword.

In this example, if the current scenario is Budget or Forecast, Oracle Hyperion Financial Management multiplies the amounts in the UnitsSold and Price accounts and inserts the product in the Sales account:

```
If HS.Scenario.Member = "Budget" Or HS.Scenario.Member = "Forecast" Then

    HS.EXP "A#Sales = A#UnitsSold * A#Price"

End If
```

## Performing Different Actions for Different Members

You can have a rule perform different actions for different members of a dimension. For example, you might want one calculation to occur if the current scenario is Budget and a different calculation to occur if the current scenario is Actual.

To conditionally perform different actions, include one or more `ElseIf` statements in an `If` structure. Have each `ElseIf` statement test for a different member; place the actions to be performed for a member beneath its `ElseIf` statement.

In this example, different accounts will be updated depending upon whether the current scenario is Budget or Actual:

- If the current scenario is Budget, Oracle Hyperion Financial Management multiplies the amounts in the `UnitsSold` and `Price` accounts and inserts the product in the `Sales` account.
- If the current scenario is Actual, Financial Management divides the `Sales` account's amount by the `UnitsSold` account's amount and inserts the quotient in the `Price` account.

```
If HS.Scenario.Member = "Budget" Then

    HS.EXP "A#Sales = A#UnitsSold * A#Price"

    ElseIf HS.Scenario.Member = "Actual" Then

        HS.EXP "A#Price = A#Sales / A#UnitsSold"

End If
```

### Tip:

To have an action occur if none of the specified conditions in the `If...Then` and `ElseIf` statements are met, VBScript enables you to add an `Else` statement to an `If` structure. See Microsoft's VBScript documentation for details.

## Setting Opening Balances of All Accounts

To set the opening balances of accounts, use the `Exp` and `IsFirst` functions in an `If` structure. `Exp` and `If` structures are introduced in the previous sections; `IsFirst` determines whether the current period is the first period in the default frequency of the current scenario. For example, if a scenario has a default frequency of `Monthly`, `IsFirst` determines whether the current period is the first month in the year.

To set opening balances, place `IsFirst` in an `If` structure's `If...Then` statement, then place `Exp` between this statement and the `End If` statement. While you can include a specific account in `Exp`'s argument, you probably will want to set the opening balances of all the accounts.

This example shows you how to set the opening balances of all accounts. You can just retype or copy this example into a Calculation rule without modifications:

```
HS.EXP "A#ALL = A#ALL.P#Prior"
```

 **Note:**

In this example, *A#* is followed by the keyword `ALL`; this means that the rule applies to all accounts. In addition, the *P#* characters are followed by the keyword `Prior`; this means that `EXP` gets the account data from the period prior to the current period.

## Creating Rules Files

You can create rules in a text editor such as Notepad ++. Rules files can be in an ASCII format that supports multibyte character sets (MBCS), or a file encoded with Unicode format, using Little Endian byte ordering. You use Visual Basic Script functions and Oracle Hyperion Financial Management functions in rules files. By default, rules files use the RLE file extension.

You can include all types of Financial Management rules in any order in the rules file. [Table 1](#) lists the basic syntax to define each routine.

**Table 10-3 Rules Routines**

Rule Routine	Syntax
Sub Calculate	Sub Calculate() 'Type your Calculation rule here. End Sub
Sub Translate	Sub Translate() 'Type your Translation rule here. End Sub
Sub Allocate	Sub Allocate() 'Type your Allocation rule here. End Sub
Sub Input	Sub Input() 'Type your Input rule here. End Sub
Sub NoInput	Sub NoInput() 'Type your NoInput rule here. End Sub
Sub Consolidate	Sub Consolidate() 'Type your Consolidation rule here. End Sub
Sub Dynamic	Sub Dynamic() 'Type your Dynamic rule here. End Sub

**Table 10-3 (Cont.) Rules Routines**

Rule Routine	Syntax
Sub Transactions	Sub Transactions() 'Type your Transactions rule here. End Sub
Sub EquityPickup	Sub EquityPickUp() 'Type your Equity Pickup rule here. End Sub
Sub OnDemand	Sub OnDemand_<ruleName> 'Type your OnDemand rule here. End Sub

**Equity Pickup Rules Example**

The following section shows a sample Equity Pickup Rules section. To calculate Equity Pickup, the application administrator must create a new section in the Rules file named Sub EquityPickUp, where EPU calculations are defined. The default Point of View when the section is run is as follows:

- Current Scenario, Year, and Period
- Entity: Owner of the pair processed
- Value: Entity currency

```
Sub EquityPickUp()

Owner = Hs.Entity.Member

Owned = Hs.Entity.Owned

OwnerDefaultCurrency = HS.Entity.DefCurrency("")

lPown = Hs.GetCell("E#" & Owned & ".I#" & Owner &
".V#[None].A#[Shares%Owned].C1#[None].C2#[None].C3#[None].C4#[None]")

Hs.Clear "A#IncomeFromSubsidiary.I#" & Owned

Hs.Exp "A#IncomeFromSubsidiary.I#" & Owned & " = E#" & Owned & ".V#" &
OwnerDefaultCurrency & ".A#NetIncome.I#[ICP Top] *" & lPown

End Sub
```

### OnDemand Rules Example

To run on-demand rules from data forms, the administrator must create a new section in the Rules file named Sub OnDemand, where OnDemand rules are defined.

```
Sub OnDemand_Calculation
```

```
HS.Exp "A#CogstP=15424"
```

```
HS.Exp "A#Admex=32452"
```

```
End Sub
```

Note that all HS functions that can be used in Sub Calculate (but no others) can also be used in On Demand rules. Also note that unlike Sub Calculate, data previously written to an IsCalculated data-point is not cleared when an On-demand rule is run.

## Loading Rules

Rules changes can affect data and are dependent on metadata. As a result, the rules load process applies a global lock on the Oracle Hyperion Financial Management server cluster. The rules load cannot proceed until any previously started operations of these types have finished:

- Consolidation
- Data entry
- Data, Journal or Security load
- Extract Data to Database
- Member list load
- Metadata load

The rules scan process has the same restrictions as the load process to enable the system to validate dimension members and other parameters. Rules scan and load processes are queued and started automatically after any blocking tasks are finished. Oracle recommends that you load rules during periods of light activity across the server cluster instead of, for example, during a long-running consolidation. You can check the Running Tasks page to see which processes, such as consolidation or data loads, are in progress.

After you load a rules file to an application, users using that application are notified that the system has changed and that they must log off from the application and log back on.

If the rules file contains intercompany transactions, you can verify the posted transactions in the application against new transactions in the rules file. The Sub Transactions section of the rules file defines the accounts that support intercompany transaction detail.

 **Note:**

Oracle recommends that you add Financial Management to the exceptions for your Web pop-up blocker. When you perform some tasks such as loading data, a status window pops up showing the task status. If you have a pop-up blocker enabled on your computer, the status window is not displayed.

To load rules:

1. Select **Consolidation**, then **Load**, and then **Application Elements**.
2. In the **Rules** section, enter the file name to load, or click **Browse** to locate the file.

 **Note:**

By default, rules files use the RLE file extension. The load process also accepts the TXT file extension, however, Oracle recommends that you use the RLE file extension.

3. **Optional:** Select **Check Integrity** to verify that posted intercompany transactions are valid with the statements in the Sub Transactions section of the rule file that you are loading.
4. **Optional:** Click **Scan** to verify that the file format is correct.
5. Click **Load**.

 **Tip:**

To reset the options to the default values, click **Reset**.

6. **Optional:** To download the log file, click **Download Log**. Click **Open** to display the log file, or **Save** and select a location to save the file locally.

## Extracting Rules

When you extract rules, they are saved to an ASCII file that supports multibyte character sets (MBCS). By default, rules files use the RLE file extension. After you extract rules, you can view and modify them in a text editor.

To extract rules:

1. Select **Consolidation**, then **Extract**, and then **Application Elements**.
2. In the **Rules** section, click **Extract**.
3. Follow the download instructions displayed in your browser to download the extracted file.  
The instructions vary depending on the Web browser that you are using. Make sure to save the file in the Web directory that you set up.
4. **Optional:** To download the log file, click **Download Log**. Click **Open** to display the log file, or **Save** and select a location to save the file locally.



## Rules Profiling

You can use rules profiling to capture in-depth information about rules execution at the major subroutine level. Rules profiles include statistics about time spent calculating, translating, and consolidating each entity during a consolidation task. You can view rule profiles to ensure that rules are run efficiently, and to address any issues that occur during rules execution or entity processing during consolidation.

You must have Application Administrator security access to run and view rules profiles.

### Adding Profile Information to Rule Files

To capture specific profiling data, you can define a block within a rule file.

To add profile information to a rule file:

1. Open a rule file and update the major subroutine with Block level information as follows:
  - Use `HS.StartTimer` to define the start of the block.
  - Use `HS.EndTimer` to define the end of the block.
  - Define a unique timer ID in the rule file; use any positive integer.
  - Define the user action, such as Calculate. The block name is appended with `_TimeID`.

Example:

```
HS.StartTimer 1, "Calculate", "", "", "" '1: Unique Timer ID, Calculate:  
User Defined Block Name
```

```
If Hs.Scenario.Member ="BudV1" or Hs.Scenario.Member="BudV2" Then
```

```
If HS.Entity.IsBase "Geographical", Then
```

```
...
```

```
HS.EndTimer 1 '1: Corresponding Unique Timer ID of earlier User defined  
block to represent End of the Block
```

2. Load the updated rule file.
3. Run the action that you want to profile.

### Starting and Stopping Rules Profiling

By default, rules profiling is not enabled. You must enable it for applications.

You can enable profiling per application across all servers, but not for individual servers.

To start rules profiling:

1. Select **Navigate**, then **Administer**, and then **Consolidation Administration**.

2. Select **Administration**, and then **Applications**, or from **Admin Tasks**, select **Applications**.
3. Select an application, right-click and then select **Profile Summary**.  
If profiling has not been run on the application, the Profile Summary page does not display any results. If profiling has been run previously, you can view the results from prior runs.
4. From the top of the screen, select **Start**.
5. The system displays the following confirmation: "It is recommended not to enable profiling in production systems with active users. System performance can be negatively affected if profiling is enabled. Do you want to enable profiling?" From the confirmation prompt, click **Yes**.

 **Note:**

If you want to stop profiling that is already started, click **Stop**.

## Viewing Rules Profile Results

When a user performs an action that invokes rules, the profile captures information about the rules execution. These actions include processes available from data grids, data forms, and the Equity Pickup module. Actions include the following: Calculate, Consolidate, Allocate, Translate, Input, No Input, Dynamic, Transaction, and Equity Pickup.

From the Profile Summary, you can view results in the Rules View or Entity View. The Rules View displays the major rules that were run.

The Entity View displays the entities based on the entity structure. It is only available if Consolidation rules were run, and displays results for Consolidation, Calculation (including the child node), and Translation processes.

To view Rules Profile results:

1. Select **Navigate**, then **Administer**, and then **Consolidation Administration**.
2. Select **Administration**, and then **Applications**.
3. Select an application, right-click and then select **Profile Summary**.  
The Profile Summary page lists the rules profiles, including the task, user, and date and time that the rules were run. If multiple profiles exist, from the dropdown, select the profile that you want to view.
4. Select **Rules View** or **Entity View**.
5. To refresh the results page, click **Refresh**.

## Rules View

### Per User Action

- Total time spent for this user action
- Amount of time spent processing rules (as a percentage of the total system time)

### Per Major Subroutine

- Sub/Block - The subroutine that was executed

- Total Invocations - Number of times the user-defined major subroutine/block in the rule was executed for the user action
- Average Time - Average time spent executing each subroutine. It is the sum of all individual elapsed times, divided by the total of the number of invocations for the sub/block.
- Total Time - Sum of all individual elapsed subroutine times. For example, if the Calculate major subroutine was run for ten entities in parallel, and each took one second, the total time would be ten seconds.
- POV Details - Top Entities by time spent (sorted from highest to lowest time) in executing the subroutine

Sub/Block	Total Invocations	Average Time	Total Time	Entities	Entity	Period	Value	Elapsed Time
Calculate	33	166	5506	UKSales	February	GBP Total	494.0	
Consolidate	8	934	7474	UKSales	February	[Proportion]	454.0	
CustomLogic	1	1	1					
Translate	1	250	250					

## Entity View

The Entity View provides information about user-initiated consolidation tasks, such as Consolidate All, Consolidate All with Data, Consolidate Impacted, and Calculate Contribution. From the Entity bar graph, you can hover over a row to see the block timing percentages. You can collapse and expand the entity rows.

### Per User Consolidation Tasks

Total time spent for the consolidation task

### Per Entity

Block	Cumulative Time	Operations	Entity	Period	Value	Thread Time	Timestamp
Base Level Calc	2652	Calculate Child Nodes	WestRegion	January		624	Wed Sep 02 15:21:36 EDT
UnitedStates	5335	Calculate Entity	WestRegion	January	USD	46	Wed Sep 02 15:21:36 EDT
WestRegion	1622	Calculate Entity	WestRegion	January	USD Total	15	Wed Sep 02 15:21:37 EDT
EastRegion	5038	Consolidate Node	WestRegion	January		218	Wed Sep 02 15:21:37 EDT
EastAdmin	187						
EastSales	218						
EastProduct	218						
A	280						
B	452						
UKSales	858						

Time spent in consolidating each branch in the entity hierarchy. You can expand entities in the hierarchy to the base-level entities. Entities with multiple parents are displayed as duplicated entries.

- Cumulative Time - The time in the left panel indicates the total time spent for consolidating branches in the entity hierarchy. The value shown for each entity is the real clock time spent processing the entity and its descendants. Due to the multi-thread nature of consolidation, the sum of individual total time from the child entities may be more than the total time from the parent entity.

- Operations- Each row in the right panel shows a separate thread that performs a subtask during the processing of the entity highlighted in the left panel.
- Thread Time - Amount of time for each thread to complete each subtask

## Deleting Rules Profiles

You can delete rules profile results for specific actions such as Consolidate or Translate, or delete all the results for all of the actions for the application.

To delete Rules Profile results:

1. Select **Navigate**, then **Administer**, and then **Consolidation Administration**.
2. Select **Administration**, and then **Applications**.
3. Select an application, right-click and then select **Profile Summary**.
4. Do one of the following tasks:
  - To delete results for a specific action, from the drop-down list, select an action and click the **Delete** icon next to it.
  - To delete all results, click **Delete All** from the top menu.
5. From the confirmation prompt, click **Yes**.

# 11

## Rule Functions

You can write these rule types using these functions:

- Allocation rules
- Calculation rules
- Consolidation rules
- Dynamic calculation rules
- Equity Pickup rules
- Input rules
- NoInput rules
- On-Demand rules
- Transactions rules
- Translation rules

Some functions are unique to specific routines, while others can be used with multiple types of rules within multiple routines.

For an overview of all functions, see [Functions Overview](#).

## Functions Overview

[Table 1](#) summarizes the Oracle Hyperion Financial Management functions, the objects with which they can be used, and the type of rules in which they can be used. Functions are listed alphabetically. Detailed sections for each function are provided after the table.



### Note:

Legacy applications migrated from Financial Management releases prior to 11.1.2.2 can still use the "Custom1...4" objects. All applications created in 11.1.2.2 should use the new `Custom(Alias)` object syntax to specify which Custom dimension is being referenced using the Custom Alias/short label/long label.

Rules types are abbreviated in this table as follows:

- Alloc - Allocation
- Calc - Calculation
- Con - Consolidation
- Dyn - Dynamic Calculation
- EPU - EquityPickUp
- Tran - Translation

- Trans - Transactions

**Table 11-1 Financial Management Functions**

Function	Description	Objects	Types of Rules
<a href="#">ABSExp</a>	Executes a calculation expression and stores the result as an absolute value.	HS	Calc, Tran, Alloc
<a href="#">AccountType</a>	Gets the account type for the member.	Account	Calc, Tran, Con, Alloc
<a href="#">AccountTypeID</a>	Gets the account type ID for the member.	Account	Calc, Tran, Con, Alloc
<a href="#">AddEntityToList</a>	Adds the specified entity and parent to a member list.	HS	Member List
<a href="#">AddEntityToListUsingIDs</a>	Using entity and parent IDs, adds the specified entity and parent to an internal list.	HS	Member List
<a href="#">AddMemberToList</a>	Adds the member to the member list.	HS	Member List
<a href="#">AddMemberToListUsingIDs</a>	Using member IDs, adds the specified member to an internal list.	HS	Member List
<a href="#">Alloc</a>	Allocates data to a cell.	HS	Alloc
<a href="#">AllowAdjFromChildren</a>	Determines if journal postings from children are allowed for the member.	Entity, Parent	Calc, Tran, Con, Alloc
<a href="#">AllowAdjs</a>	Determines if journal postings are allowed for the member.	Entity, Parent	Calc, Tran, Con, Alloc
<a href="#">ApplicationName</a>	Returns the name of the application in which rules are running.	AppSettings	Calc, Tran, Con, Alloc
<a href="#">CalculateExchangeRate</a>	Calculates the exchange rate from one currency to another.	HS	Calc, Tran, Con, Alloc, Dyn, Trans
<a href="#">CalculateRate</a>	Gets the current exchange rate for the specified entity.	HS	Calc, Tran, Con, Alloc, Dyn, Trans
<a href="#">CalcStatus</a>	Gets the calculation status for the cell.	HS	Calc, Tran, Con, Alloc
<a href="#">Clear</a>	Removes data from a cell.	HS	Calc, Tran, Con, Alloc
<a href="#">Con</a>	Puts data into the [Proportion] and [Elimination] accounts.	HS	Con
<a href="#">Consol1, Consol2, Consol3</a>	Gets the Consol1...3 system account value for the node.	Node	Calc, Tran, Con, Alloc

**Table 11-1 (Cont.) Financial Management Functions**

Function	Description	Objects	Types of Rules
<a href="#">ConsolidateYTD</a>	Determines if the scenario is consolidated using the YTD or periodic method.	Scenario	Calc, Tran, Con, Alloc
<a href="#">ContainsCellText</a>	Determines if the cell contains cell text.	HS	Calc, Tran, Con, Alloc
<a href="#">ContainsCellTextWithLabel</a>	Determines if the cell contains cell text for the specified label.	HS	Calc, Tran, Con, Alloc
<a href="#">Currency</a>	Gets the currency type for the application or the value member.	AppSettings, Value	Calc, Tran, Con, Alloc
<a href="#">CustomTop</a>	Gets the CustomTopMember for the account.	Account	Calc, Tran, Con, Alloc
<a href="#">DataUnitItem</a>	Gets the data unit item to process during consolidation, calculation, or translation.	HS	Calc, Tran, Con
<a href="#">Decimal</a>	Gets the number of decimal places for the specified account.	Account	Calc, Tran, Con, Alloc
<a href="#">DefaultFreq</a>	Gets the default scenario frequency.	Scenario	Calc, Tran, Con, Alloc
<a href="#">DefaultParent</a>	Gets the default parent for the member.	Account	Calc, Tran, Con, Alloc
<a href="#">DefaultTranslate</a>	Calculates translation by bypassing rules.	HS	Tran
<a href="#">DefaultView</a>	Gets the default scenario view.	Scenario	Calc, Tran, Con, Alloc
<a href="#">DefCurrency</a>	Gets the default currency for the entity or parent.	Entity, Parent	Calc, Tran, Con, Alloc
<a href="#">Down</a>	Gets the percentage of Direct Ownership (DOWn) for the node.	Node	Calc, Tran, Con, Alloc
<a href="#">Dynamic</a>	Specifies the formula for the dynamic accounts that need calculations.	HS	Dyn
<a href="#">Exp</a>	Executes a calculation expression and puts data into a specified point of view.	HS	Calc, Tran, Alloc
<a href="#">GetCell</a>	Gets the data contained in a cell.	HS	Calc, Tran, Con, Alloc
<a href="#">GetCellNoData</a>	Gets the data contained in a cell and indicates if the cell contains no data.	HS	Calc, Tran, Con, Alloc

**Table 11-1 (Cont.) Financial Management Functions**

Function	Description	Objects	Types of Rules
<a href="#">GetCellRealData</a>	Gets the data contained in a cell and indicates if the cell contains real data.	HS	Calc, Tran, Con, Alloc
<a href="#">GetCellText</a>	Gets the default cell text for a specified Point of View.	HS	Calc, Tran, Con, Alloc
<a href="#">GetCellTextWithLabel</a>	Gets the cell text from the Point of View for the specified cell text label.	HS	Calc, Tran, Con, Alloc
<a href="#">GetCellType</a>	Gets the cell type.	HS	Calc, Tran, Con, Alloc
<a href="#">GetCustomLabelArray</a>	Returns a list of Custom dimension names and aliases.	HS	Calc, Tran, Con, Alloc
<a href="#">GetItem</a>	Gets an individual record to process for consolidation.	DataUnit	Calc, Tran, Con
<a href="#">GetItemIDs2</a>	Gets an individual record to process for consolidation using dimension ID numbers.	DataUnit	Calc, Tran, Con
<a href="#">GetNumItems</a>	Gets the number of records to process for consolidation.	DataUnit	Calc, Tran, Con
<a href="#">GetNumLID</a>	Gets the number of line-item details for the specified POV.	HS	Calc, Tran, Con, Alloc
<a href="#">GetRate</a>	Gets the currency rate for a cell.	HS	Calc, Tran, Con, Alloc
<a href="#">GetSubmissionGroup</a>	Gets the submission group for a cell.	HS	Calc, Tran, Con, Alloc
<a href="#">GetSubmissionPhase</a>	Gets the submission phase for a cell.	HS	Calc, Tran, Con, Alloc
<a href="#">Holding</a>	Gets the holding company for the member.	Entity, Parent	Calc, Tran, Con, Alloc
<a href="#">ICPTopMember</a>	Gets the ICPTopMember of the current Account dimension member or the specified account member.	Account	Calc, Tran, Con, Alloc
<a href="#">ICPWeight</a>	Gets the percentage of ICP entity balances that aggregate to the [ICP Top] value member.	AppSettings	Calc, Tran, Con, Alloc
<a href="#">IDFromMember</a>	Gets the ID number for the specified member.	Account, Custom1 through Custom4, Entity, ICP, Parent, Year, Period, Scenario, Value, View	Calc, Tran, Con, Alloc



**Table 11-1 (Cont.) Financial Management Functions**

Function	Description	Objects	Types of Rules
ImpactStatus	Changes the status of the specified data unit to impacted.	HS	Calc
Input	Enables input at parent entity level for specified account.	HS	Input
IsAlmostEqual	Checks to see if two values are equal.	HS	Calc, Tran, Con, Alloc, Dyn, Trans
IsBase	Determines if the member is a base member.	Account, Custom1 through Custom4, Entity, Parent, Node	Calc, Tran, Con, Alloc
IsCalculated	Determines if the account is calculated.	Account	Calc, Tran, Con, Alloc
IsChild	Determines if the member is a child of another member.	Account, Custom1 through Custom4, Entity, Parent, Node	Calc, Tran, Con, Alloc
IsConsolidated	Determines if the account is consolidated.	Account	Calc, Tran, Con, Alloc
IsDescendant	Determines if the member is a descendant of another member.	Account, Custom1 through Custom4, Entity, Parent, Node	Calc, Tran, Con, Alloc
IsFirst	Determines if the period or year is the first for the application.	Period, Year	Calc, Tran, Con, Alloc
IsICP	Determines if the member is an ICP.	Account, Entity, Parent	Calc, Tran, Con, Alloc
IsLast	Determines if the year or period is the last for the application.	Period, Year	Calc, Tran, Con, Alloc
IsTransCur	Determines if the value member is a translated currency member.	Value	Calc
IsTransCurAdj	Determines if the value member is a translated currency Adj member.	Value	Calc
IsValidDest	Determines if the specified POV is a valid destination.	HS	Calc, Tran, Con, Alloc
IsZero	Checks to see if the passed in value is zero.	HS	Calc, Tran, Con, Alloc, Dyn, Trans
List	Gets the elements contained in a list.	Account, Custom1 through Custom4, Entity, Parent, ICP, Node, Scenario	Calc, Tran, Con, Alloc
Member	Gets the member name.	Entity, Parent, Period, Scenario, Value, Year, View	Calc, Tran, Con, Alloc

**Table 11-1 (Cont.) Financial Management Functions**

Function	Description	Objects	Types of Rules
MemberFromID	Gets the member for the specified ID number.	Account, Custom1 through Custom4, Entity, ICP, Parent, Year, Period, Scenario, Value, View	Calc, Tran, Con, Alloc
Method	Gets the consolidation method for the member.	Node	Calc, Tran, Con, Alloc
NoInput	Prevents input into cells.	HS	Noinput
NoRound	Turns off rounding.	HS	Calc, Tran, Con, Alloc
NumBase	Gets the number of base members.	Account, Custom1 through Custom4, Entity, Parent, Node, Period	Calc, Tran, Con, Alloc
Number	Gets the current period number.	Period	Calc, Tran, Con, Alloc
NumChild	Gets the number of children for the member.	Account, Custom1 through Custom4, Entity, Parent, Node	Calc, Tran, Con, Alloc
NumCustom	Gets the number of Custom dimensions defined for the application.	HS	Calc, Tran, Con, Alloc
NumDescendant	Gets the number of descendants for the member.	Account, Custom1 through Custom4, Entity, Parent, Node	Calc, Tran, Con, Alloc
NumPerInGen	Gets the number of periods in the generation for the current period being processed.	Period	Dynamic
NumPeriods	Gets the number of periods defined for the frequency of the specified scenario.	Scenario	Dynamic
OpenCellTextUnit	Returns the cell text for multiple cells.	HS	Calc, Tran, Con
OpenDataUnit	Gets a data unit for consolidation.	HS	Calc, Tran, Con
OpenDataUnitSorted	Gets the data units for calculation, translation, or consolidation, in sorted order.	HS	Calc, Tran, Con
Owned	Gets the Owned entity of the pair currently processed.	Entity	Equity Pickup
Owner	Gets the Owner entity of the pair currently processed.	Entity	Equity Pickup
PCon	Gets the percentage of consolidation.	Node	Calc, Tran, Con, Alloc

**Table 11-1 (Cont.) Financial Management Functions**

Function	Description	Objects	Types of Rules
PEPU	Gets the percentage of ownership from the EPU table.	HS	Equity Pickup
PeriodNumber	Gets the period number in the view for the data that is being retrieved.	View	Dynamic
PlugAcct	Gets the plug account.	Account	Calc, Tran, Con, Alloc
POwn	Gets the percentage of ownership.	Node	Calc, Tran, Con, Alloc
PVAForBalance	Determines default translation method for BALANCE accounts.	AppSettings	Calc, Tran, Con, Alloc
PVAForFlow	Determines default translation method for FLOW accounts.	AppSettings	Calc, Tran, Con, Alloc
RateForBalance	Gets the default rate for balance.	AppSettings	Calc, Tran, Con, Alloc
RateForFlow	Gets the default rate for flow.	AppSettings	Calc, Tran, Con, Alloc
ReviewStatus	Gets the process management review status for the cell.	HS	Calc, Tran, Con, Alloc
ReviewStatusUsingPhaseID	Gets the process management review status by phase ID for the cell.	HS	Calc, Tran, Con, Alloc
Round	Rounds the data.	HS	Calc, Tran, Con, Alloc
Scale	Gets the scale of the specified currency.	Currency	Calc, Tran, Con, Alloc
SecurityAsPartner	Gets the security class for the ICP entity.	Entity, Parent	Calc, Tran, Con, Alloc
SecurityClass	Gets the security class for a dimension member.	Account, Scenario, Entity, Parent, Custom1 through Custom4	Calc, Tran, Con, Alloc
SetCellTextWithLabel	Writes a text string to cell text for a specified POV and cell text label.	HS	Calc
SetData	Sets an individual record.	HS	Calc, Tran
SetDataWithPOV	Inserts data into the node or currency cube.	HS	Calc, Tran
SubmissionGroup	Gets the process management submission group for a dimension member.	HS	Calc, Tran, Con, Alloc
SupportsProcessManagement	Determines if a scenario supports process management.	Scenario	Calc, Tran, Con, Alloc

**Table 11-1 (Cont.) Financial Management Functions**

Function	Description	Objects	Types of Rules
<a href="#">SupportsTran</a>	Defines the accounts in the application that require Intercompany Transaction detail support.	Scenario, Year, Entity, Account, C1...4	Tran
<a href="#">SwitchSign</a>	Determines if credits are switched to debits for the member.	Custom1 through Custom4	Calc, Tran, Con, Alloc
<a href="#">SwitchType</a>	Determines if account types are switched for the member.	Custom1 through Custom4	Calc, Tran, Con, Alloc
<a href="#">Trans</a>	Translates using YTD method.	HS	Tran
<a href="#">TransPeriodic</a>	Translates using periodic method.	HS	Tran
<a href="#">UD1...3</a>	Gets the user-defined attribute for the member.	Account, Entity, Parent, Scenario, Custom1 through Custom4	Calc, Tran, Con, Alloc
<a href="#">ValidationAccount</a>	Gets the validation account.	AppSettings	Calc, Tran, Con, Alloc
<a href="#">ValidationAccountEx</a>	Gets the validation account for the process management submission phase.	AppSettings	Calc, Tran, Con, Alloc
<a href="#">XBRLTags</a>	Gets the XBRL tag for the account.	Account	Calc, Tran, Con, Alloc

## ABSExp

Executes a calculation expression and stores the result as an absolute value. This function is the same as the Exp function except that it stores the resulting value as an absolute value. This function can be used in these types of rules:

- Calculation
- Translation
- Allocation

### Syntax

```
HS.ABSExp "DestPOV = Expression"
```

**Table 11-2 Syntax for ABSExp Function**

Parameter	Description
<i>DestPOV</i>	<p>A destination point of view that identifies where to put the data</p> <p>You must specify an Account member, and you can optionally specify ICP and Custom members. Note these usage rules:</p> <ul style="list-style-type: none"> <li>• If you do not specify an ICP member, the default is [ICP None].</li> <li>• To avoid populating the database with unwanted values, rules should be as explicit as possible in terms of defining where data should reside. A good practice is to include clearly defined Custom dimension intersections for the Account dimension. Such clearly defined intersections utilize Oracle Hyperion Financial Management validation checks to avoid writing data to invalid intersections. See <a href="#">Dimension Intersection Considerations</a>.</li> </ul>
<i>Expression</i>	A calculation expression

**Return Value**

None.

**Example**

This example sets the amount in the StateTax account. The example calculates the absolute amount by multiplying the amount in the Sales account for 2014 by the rate in the StateRate account for 2014.

```
HS.ABSExp "A#StateTax = A#Sales.Y#2014 * A#StateRate.Y#2014"
```

## AccountType

Gets the account type for the current Account member or for a specified account. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**

```
HS.Account.AccountType("Account")
```

```
HS.Account.AccountType("")
```

where, *Account* is the name of a valid Account member.

You can use a blank string (" ") to apply this function to the current member only if you are using the function in the Sub Consolidate subroutine. Otherwise, specify an account when using this function.

### Return Value

The account type for the specified account.



#### Note:

Account types use all capital letters.

Valid account types are:

- ASSET
- LIABILITY
- REVENUE
- EXPENSE
- DYNAMIC
- FLOW
- BALANCE
- BALANCERECURRING
- CURRENCYRATE
- GROUPLABEL

### Example

In this example, if the account type for the Sales account is REVENUE, then statements between the If...Then and End If statements are executed.

```
If HS.Account.AccountType("Sales") = "REVENUE" Then
```

```
...
```

```
End If
```

## AccountTypeID

Gets the account type ID for the current account member or for a specified account. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation

- Allocation
- Dynamic Calculation
- Transactions

### Syntax

```
HS.Account.AccountTypeID("Account")
```

```
HS.Account.AccountTypeID("")
```

where *Account* is the name of a valid Account member

You can use a blank string (" ") to apply this function to the current member only if you are using the function in the Sub Consolidate subroutine. Otherwise, you must specify an account when using this function.

### Return Value

The ID for the specified account. [Table 1](#) lists valid account types with corresponding IDs.

**Table 11-3 Account Type IDs**

Account Type	ID
REVENUE (INCOME in previous releases)	0
EXPENSE	1
ASSET	2
LIABILITY	3
BALANCE	4
FLOW	5
CURRENCYRATE	7
GROUPLABEL	10
BALANCERECURRING	11
DYNAMIC	12

### Example

```
If HS.Account.AccountTypeID("Investments") = 2 Then
```

```
...
```

```
End If
```

## AddEntityToList

Adds the specified member to a member list. This function can be used only in the Member Lists file.

**Syntax**

```
HS.AddEntityToList("Parent", "Child")
```

**Table 11-4 Syntax for AddEntityToList Function**

Parameter	Description
<i>Parent</i>	Name of a valid parent entity.
<i>Child</i>	Name of a child of the parent entity.

**Return Value**

None

**Example**

```
HS.AddEntityToList "UnitedStates", "Maine"
```

## AddEntityToListUsingIDs

Using the ID for the entity and parent, adds the specified member to a member list. This function can be used only in the Member Lists file.

**Syntax**

```
HS.AddEntityToListUsingIDs(ParentID, ChildID)
```

**Table 11-5 Syntax for AddEntityToListUsingIDs Function**

Parameter	Description
<i>ParentID</i>	The ID for the parent entity.
<i>ChildID</i>	The ID for the child of the parent entity.

**Return Value**

None

## AddMemberToList

Adds the specified member to a member list. This function can be used only in the Member Lists file.

**Syntax**

```
HS.AddMemberToList("Member")
```

where *Member* is the name of a valid dimension member.



**Return Value**

None

**Example**

```
HS.AddMemberToList "July"
```

## AddMemberToListUsingIDs

Adds the specified member to a member list. This function can be used only in the Member Lists file.

**Syntax**

```
HS.AddMemberToListUsingIDs (MemberID)
```

where *MemberID* is the ID for a valid dimension member.

**Return Value**

None

## Alloc

Allocates data from one point of view to another. This function can be used in Allocation rules.

**Syntax**

```
HS.Alloc ("SourcePOV", "DestPOV", "EntityList", "AllocExp", "PlugAccount")
```

**Table 11-6 Syntax for Alloc Function**

Parameter	Description
<i>SourcePOV</i>	<p>A source point of view for the data that is being allocated.</p> <p>You must specify an Account member, and you can optionally specify ICP and Custom members. If you do not specify ICP and Custom members:</p> <ul style="list-style-type: none"> <li>• The default ICP member is ICP Top.</li> <li>• The default Custom member is TopMember for that account.</li> </ul>

Table 11-6 (Cont.) Syntax for Alloc Function

Parameter	Description
<i>DestPOV</i>	<p>A destination point of view that identifies where to allocate the data.</p> <p>You must specify an Account member, and you can optionally specify ICP and Custom members. Note these usage rules:</p> <ul style="list-style-type: none"> <li>• If you do not specify an ICP member, the default is [ICP None].</li> <li>• If you do not specify Custom members, the default is [None].</li> <li>• If you specify an Entity member, it is used as the parent if the <i>EntityList</i> argument is [Base].</li> </ul>
<i>EntityList</i>	<p>A member list that identifies the entities to which the data is be allocated. You can use the system-defined [Base] entity list or you can use a user-defined list.</p> <p>If you use the [Base] system-defined list, the system uses the entity specified in the destination point of view as the parent member. If you use a user-defined list and that list contains parent members, they are skipped.</p>
<i>AllocExp</i>	<p>An expression that identifies the data to be allocated to each entity. This expression can contain these types of values:</p> <ul style="list-style-type: none"> <li>• Numbers</li> <li>• Account Expressions that identify a numeric value. You can specify an Account member, and you can optionally specify members of the ICP and Custom dimensions. Note these usage rules: <ul style="list-style-type: none"> <li>If you do not specify Custom members, the default is TopMember.</li> <li>If you do not specify an ICP member, the default is ICP Top.</li> <li>If you do not specify Scenario, Year, Period, View, or Value members, the default is Current.</li> <li>If you do not specify an Entity member, the default is the destination entity.</li> </ul> </li> </ul>

**Table 11-6 (Cont.) Syntax for Alloc Function**

Parameter	Description
<i>PlugAccount</i>	<p>The name of a plug account. This argument is optional, and is used to reverse the source point of view amount, taking into consideration the attribute of the source point of view account versus the plug account.</p> <p>You must specify an Account member, and you can optionally specify ICP and Custom members. Note these usage rules:</p> <ul style="list-style-type: none"> <li>You cannot use this argument if the source point of view Entity member is a parent, or if the source point of view Value member is not Entity Currency.</li> <li>If you do not specify an ICP member, the default is ICP None.</li> <li>If you do not specify Custom members, the default is None.</li> </ul>

**Return Value**

None.

**Example**

In this example, data from the TangibleAssets account is allocated to the Cash account.

```
Call HS.ALLOC
("A#TangibleAssets", "A#Cash", "NewEngland", "A#TangibleAssets", "A#Plug")
```

## AllowAdjFromChildren

Specifies if journal postings from children are allowed for the specified entity or parent member. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

 **Note:**

A member is a child if it is one level directly below a member in a tree hierarchy.

## Syntax

```
HS.Entity.AllowAdjFromChildren("Entity")
```

```
HS.Entity.AllowAdjFromChildren(" ")
```

```
HS.Parent.AllowAdjFromChildren("Entity")
```

```
HS.Parent.AllowAdjFromChildren(" ")
```

where *Entity* is the name of a valid Entity or Parent member.

Use a blank string (" ") to apply this function to the current entity or parent.

## Return Value

A Boolean expression that is True if journal postings from children are permitted for the specified entity, False if journal postings from children are not permitted.

## Example

In this example, if journal postings from children of France are allowed, then statements between the If...Then and End If statements are executed.

```
If HS.Entity.AllowAdjFromChildren("France") = TRUE then
```

```
...
```

```
End If
```

# AllowAdjs

Specifies if journal postings are allowed for the specified entity or parent member. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**

```
HS.Entity.AllowAdjs("Entity")
```

```
HS.Entity.AllowAdjs(" ")
```

```
HS.Parent.AllowAdjs("Entity")
```

```
HS.Parent.AllowAdjs(" ")
```

where *Entity* is the name of a valid Entity or Parent member.

Use a blank string (" ") to apply this function to the current entity or parent.

**Return Value**

A Boolean expression that is True if journal postings are permitted for the specified entity, False if journal postings are not permitted.

**Example**

In this example, if journal postings for France are allowed, then statements between the If... Then and End If statements are executed.

```
If HS.Entity.AllowAdjs("France") = TRUE then
```

```
...
```

```
End If
```

## ApplicationName

Returns the name of the application in which rules are running. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**

```
HS.ApplicationName
```

**Return Value**

Name of the application in which rules are running.

**Example**

```
strApp=HS.ApplicationName
```

## CalculateExchangeRate

Calculate the exchange rate from one currency to another.

**Syntax**

```
HS.CalculateExchangeRate (ScenarioID, YearID, PeriodID, EntityID,
RateAccountID, FromCurrencyID, ToCurrencyID)
```

**Table 11-7 Syntax for CalculateExchangeRate Function**

Parameter	Description
<i>ScenarioID</i>	The scenario ID.
<i>YearID</i>	The year ID.
<i>PeriodID</i>	The period ID.
<i>EntityID</i>	The entity ID.
<i>RateAccountID</i>	The rate account ID.
<i>FromCurrencyID</i>	The ID for the currency that you are going from.
<i>ToCurrencyID</i>	The ID for the currency that you are going to.

**Return Value**

The exchange rate.

## CalculateRate

Gets the current exchange rate for the specified entity.

**Syntax**

```
HS.CalculateRate (ScenarioID, YearID, PeriodID, EntityID, ValueID,
RateAccountID)
```

**Table 11-8 Syntax for CalculateRate Function**

Parameter	Description
<i>ScenarioID</i>	The scenario ID.
<i>YearID</i>	The year ID.
<i>PeriodID</i>	The period ID.
<i>EntityID</i>	The entity ID.
<i>ValueID</i>	The value ID.
<i>RateAccountID</i>	The rate account ID.

**Return Value**

The exchange rate for the specified entity.

## CalcStatus

Gets the calculation status for the specified point of view. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**

```
HS.CalcStatus("Scenario.Year.Period.Entity.Value")
```

**Table 11-9 Syntax for CalcStatus Function**

Parameter	Description
<i>Scenario</i>	Name of a valid Scenario member.
<i>Year</i>	A valid year.
<i>Period</i>	A valid period.
<i>Entity</i>	Name of a valid Entity member.
<i>Value</i>	Name of a valid Value member.

**Return Value**

A string that contains the calculation status for the specified point of view. Valid status codes are listed below.

**Table 11-10 Calculation Status Codes**

Status Code	Description
OK	None of the data for the specified Scenario, Year, Period, Entity, and Value dimensions has changed.
OK ND	OK - No Data. The calculation has been effectively run, but calculations were not run for no data.
OK SC	OK - System changed. A change has taken place that may affect the data for the specified Scenario, Year, Period, Entity, and Value dimensions. For example, a new rules file, metadata file, or member list file has been loaded, or the currency rate has changed. The data itself, such as a value of 10,000 has not changed. Rather, some change has taken place, perhaps in a dimension member attribute. For example, the number of decimals associated with the account has been set to two, requiring the value to be changed to 10,000.00.

Table 11-10 (Cont.) Calculation Status Codes

Status Code	Description
CH	Needs Calculation. At least one data cell for the specified Scenario, Year, Period, Entity, and Value dimensions has been changed, or metadata parameters or rules have changed. As a result, other data cells in this dimension may not be current because calculation has not been run. For base-level entities, you may have entered the data cell through data entry or by a data file load. For any entity, the data cell may have been entered by a journal posting.
CH ND	Needs Calculation - No Data. This indicates the first time that calculation will be performed on the cell.
TR	Needs Translation. The selected Value dimension member is not the entity's default currency, and its translated values may not be up to date.
TR ND	Needs Translation - No Data. This indicates the first time that translation will be performed on the cell.
CN	Needs Consolidation. The data for the specified Scenario, Year, Period, Entity, and Value dimensions may not be current because any of the following has changed: <ul style="list-style-type: none"> <li>• Data for a child entity</li> <li>• Data for the same entity's default currency</li> <li>• Metadata parameters or rules</li> </ul>
CN ND	Needs Consolidation - No Data. The parent has no data, but data for a child entity has changed. This indicates the first time that consolidation will be performed on the cell.
Locked	The data for the specified Scenario, Year, Period, Entity, and Value dimensions has been locked by an administrator. It can no longer be modified manually or through calculations. You can use the <a href="#">Alloc</a> function to modify data in a locked destination POV.
NoData	No data exists for the specified Scenario, Year, Period, Entity, and Value dimensions.
NoAccess	The user does not have rights for the specified dimension member.



### Example

In this example, if the status for the specified point of view is "OK", then statements between the If...Then and End If statements are executed.

```
If HS.CalcStatus("S#Actual.Y#2014.P#January.E#Connecticut.  
V#<EntityCurrency>") = "OK" Then
```

```
...
```

```
End If
```

## CellTextUnitItem

Returns the cell text unit item to process during consolidation, calculation, or translation. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation

### Syntax

```
HS.OpenCellTextUnit("", "[Default]", "Entity", "Ascending")
```

### Return Value

The entry for the specified item.

**Example**

```
Set MyCellTextUnit = HS.OpenCellTextUnit("", "[Default]", "Entity",  
"Ascending")
```

```
NumItems= MyCellTextUnit.GetNumItems
```

```
For i = 0 to NumItems - 1
```

```
Scenario = MyCellTextUnit.Item(i).Scenario
```

```
Year = MyCellTextUnit.Item(i).Year
```

```
Period = MyCellTextUnit.Item(i).Period
```

```
Entity = MyCellTextUnit.Item(i).Entity
```

```
Value = MyCellTextUnit.Item(i).Value
```

```
Account = MyCellTextUnit.Item(i).Account
```

```
ICP = MyCellTextUnit.Item(i).ICP
```

```
Flow = MyCellTextUnit.Item(i).Custom("Flows")
```

```
Nature = MyCellTextUnit.Item(i).Custom("Nature")
```

```
...
```

```
...
```

```
...
```

```
CellText = MyCellTextUnit.Item(i).CellText
```

```
POV = MyCellTextUnit.Item(i).POV
```

```
Next
```

## Clear

Removes data from combinations of Account, ICP, and Custom members. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.Clear "Account.ICP.Custom1.Custom2.Custom3.Custom4"
```

**Table 11-11 Syntax for Clear Function**

Parameter	Description
<i>Account</i>	Name of a valid Account member.
<i>ICP</i>	Name of a valid ICP member.
<i>Custom1, Custom2, Custom3, Custom4</i>	Name of valid Custom1, Custom2, Custom3, and Custom4 members.

To remove data from all cells that intersect the current point of view, which consists of the current Entity, Period, Scenario, Value, View, and Year members, place the ALL keyword after the A# characters as in this example:

```
HS.Clear "A#ALL"
```

To clear all intersections of cells and Custom or ICP dimensions, use the ALL keyword or omit the A# characters. This example omits the A# characters to clear data from all account intersections with the GolfBalls member of the Custom1 dimension:

```
HS.Clear "C1#GolfBalls"
```

### Return Value

None.

### Example

This example clears the data stored in the intersection of the Sales account and the GolfBalls member of the Custom1 dimension.

```
HS.Clear "A#Sales.C1#GolfBalls"
```

## Con

Puts data into the [Proportion] and [Elimination] Value dimension members. This function can be used in Consolidation rules.

## Syntax

Hs.Con (\*"DestPOV", Factor, "Nature")

**Table 11-12 Syntax for Con Function**

Parameter	Description
<i>DestPOV</i>	Combination of these elements: <ul style="list-style-type: none"> <li>• Account</li> <li>• Custom1, Custom2, Custom3, Custom4</li> <li>• Intercompany</li> <li>• Entity</li> <li>• Value</li> </ul>
<i>Factor</i>	A number or an expression using mathematical operators (+ - * /) or functions such as HS.GetCell.
<i>Nature</i>	<p>A string used for audit purposes. This string is stored in the database and provides information about the accounting purpose of the transaction.</p> <p>To allow users to view consolidation source and destination transactions after running a consolidation, you must include text in this parameter. If you do not include text, the transaction information is not stored. You can view transaction information from data grid Source or Destination Transaction options, or from the Entity Detail report.</p> <p>You can also use the Nature string information to generate journal reports for consolidation and elimination entries. If you want to see tis data in a journal report, this parameter is required. Journals for Proportion and Elimination entries are reported from RTS/RTD tables. Depending on your requirements, it may be appropriate to create the journal data for [Elimination] HS.Con entries, but not for [Proportion] HS.Con entries to reduce the volume of RTS/RTD table entries.</p> <p>Using the Nature parameter, which generates RTS/RTD table entries, increases the size of the database and may impact consolidation performance.</p>

## Return Value

None.

## Example

```
Call HS.Con ("V#[Elimination]",-1*dPCon,"")
```

```
Call HS.Con ("V#[Elimination]",-1*dPCon, "Elimination")
```

```
Call HS.Con ("V#[Elimination]"PCON, "DefaultConsolidation")
```

## Consol1, Consol2, Consol3

Gets the value in Consol1, Consol2, or Consol3 account for the specified parent.entity node. This function can be used in these types of rules:

- Calculation
- Allocation

### Syntax

Combination of scenario, year, period, and parent.entity members.

```
HS.Node.Consoln ("S#Scenario.Y#Year.P#Period.E#Parent.Entity")
```

```
HS.Node.Consoln ("")
```



### Note:

Use a blank string ("") to apply this function to the current scenario, year, period, and entity.

**Table 11-13 Syntax for Consol1, Consol2, Consol3 Functions**

Parameter	Description
<i>Scenario</i>	Name of a valid Scenario member.
<i>Year</i>	A valid year.
<i>Period</i>	A valid period.
<i>Parent.Entity</i>	Name of a valid Parent.Entity node.

### Return Value

The value in the Consol1, Consol2, or Consol3 system account.

### Example

This example gets the Consol1 value for the Group1.Ent1 node in the actual scenario.

```
dVar1 = HS.Node.Consol1 ("S#Actual.E#Group1.Ent1")
```

## ConsolidateYTD

Determines if the current Scenario member or a specified scenario member is consolidated year to date. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation

- Allocation

### Syntax

```
HS.Scenario.ConsolidateYTD("Scenario")
```

```
HS.Scenario.ConsolidateYTD("")
```

where Scenario is the name of a valid Scenario member.

Use a blank string ("") to apply this function to the current member.

### Return Value

A Boolean expression that is True if the scenario is consolidated using the year-to-date method, False if the scenario is consolidated using the periodic method.

### Example

In this example, if the Actual scenario is set up to be consolidated using the year-to-date method, then statements between the If...Then and End If statements are executed.

```
If HS.Scenario.ConsolidateYTD("Actual") = TRUE Then
```

```
...
```

```
End If
```

## ContainsCellText

Determines if the specified cell contains cell text. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.ContainsCellText("POVExpression")
```

where *POVExpression* is a combination of members. If you do not specify a dimension, these default values are used.

- Account - [none]
- ICP - [ICP None]
- Custom1...4 - [None]
- Scenario - Current Scenario member

- Entity - Current Entity member
- Value - Current Value member
- Year and Period - Current member

 **Note:**

The default entries apply when you use this function in Calculation rules. When used in Allocation, Translation, or Consolidation rules, you must specify the full sub-cube Account/ICP/Customs POV. The Scenario, Year, Period, View, Entity, and Value dimensions default to the current members.

**Return Value**

A Boolean expression that is True if the specified data cell contains cell text for any cell text label; False otherwise.

**Example**

In this example, if the specified cell does not contain cell text for any cell text label, then statements between the If...Then and End If lines are executed.

```
If HS.ContainsCellText("A#Sales.C1#Prod1.C2#Region1") = "False" then
```

```
...
```

```
End If
```

 **Note:**

If you are using cell text labels, this function returns a True value if any of the existing cell text labels (including [Default]) contain a cell text entry.

## ContainsCellTextWithLabel

Determines if the cell contains cell text for the specified label and updates the cell text information. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**

```
HS.ContainsCellTextWithLabel("POVExpression", "CellTextLabel")
```

where POVEExpression is a combination of members. If you do not specify a dimension, these default values are used:

- Account - [none]
- ICP - [ICP None]
- Custom1...4 - [None]
- Scenario - Current Scenario member
- Entity - Current Entity member
- Value - Current Value member
- Year and Period - Current member

CellTextLabel is either the default cell text label ([Default]) or one of the valid loaded cell text labels.

 **Note:**

The default entries apply when you use this function in Calculation rules. When used in Allocation, Translation, or Consolidation rules, you must specify the full sub-cube Account/ICP/Customs POV. The Scenario, Year, Period, View, Entity, and Value dimensions default to the current members.

### Return Value

A Boolean expression that is True if the specified data cell contains cell text for the cell text label specified; False otherwise.

### Example

In this example, if the specified cell does not contain cell text for the cell text label specified, then statements between the If...Then and End If lines are executed.

```
If HS.ContainsCellTextWithLabel ("A#Sales.C1#Prod1.C2#Region1", "Rating") =
"False" then ... End If
```

## Currency

Gets the currency for the application or for the current Value dimension member. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation



### Syntax

```
HS.AppSettings.Currency
```

```
HS.Value.Currency
```

### Return Value

A string that contains the currency name for the application or for the Value member.

### Example

In this example, if Euro is the currency of the application then statements between the If...Then and End If statements are executed.

```
If HS.AppSettings.Currency = "Euro" Then  
  
    ...  
  
End If
```

## CustomTop

Returns the CustomTopMember for the current or specified Account member. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.Account.CustomTop("Member"", "CustomDimName")
```

```
HS.Account.CustomTop("", "CustomDimName")
```

where *CustomDimName* is a Custom dimension, and *Member* is an Account dimension member. The member not being specified is only appropriate in the Sub Consolidate subroutine.



#### Note:

Use a blank string ("") to apply this function to the current member.

### **Return Value**

The CustomTopMember for the specified Account member.

### **Example**

```
HS.Account.CustomTop("Prod", "Sales")
```

## DataUnitItem

Returns the data unit item to process during consolidation, calculation, or translation. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation

### **Syntax**

```
HS.OpenDataUnit("")
```

### **Return Value**

The member label for the specified item.

**Example**

```
Set MyDataUnit = HS.OpenDataUnit("")

NumItems= MyDataUnit.GetNumItems

For i = 0 to NumItems - 1

Account = MyDataUnit.Item(i).Account

ICP = MyDataUnit.Item(i).ICP

Flow = MyDataUnit.Item(i).Custom("Flows")

Nature = MyDataUnit.Item(i).Custom("Nature")

...

...

...

Data = MyDataUnit.Item(i).Data

POV = MyDataUnit.Item(i).POV

Next
```

## Decimal

Gets the number of decimal places for the specified account. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

## Syntax

```
HS.Account.Decimal("AccountName")
```

```
HS.Account.Decimal("")
```



### Note:

Use a blank string ("" ) to apply this function to the current account when using the Sub Consolidate subroutine.

```
HS.Account.Decimal(Var1)
```

**Table 11-14 Syntax for Decimal Function**

Parameter	Description
<i>AccountName</i>	Name of a valid Account member
<i>Var1</i>	VBScript variable representing an Account member

## Return Value

An integer representing the decimal value assigned to the account. Valid values are 0 to 9.

## Example

In this example, if the number of decimal places assigned to the Sales account is 2, then statements between the If...Then and End If statements are executed.

```
If HS.Account.Decimal("Sales") = 2 Then
```

```
...
```

```
End If
```

## DefaultFreq

Gets the default frequency for the current Scenario member or for a specified scenario member. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

## Syntax

```
HS.Scenario.DefaultFreq("Scenario")
```

```
HS.Scenario.DefaultFreq("")
```



### Note:

Use a blank string ("") to apply this function to the current member.

where *Scenario* is the name of a valid Scenario member.

## Return Value

A string that contains the default frequency for the scenario.

## Example

In this example, if the default frequency for the Actual scenario is YTD, then statements between the If...Then and End If statements are executed.

```
If HS.Scenario.DefaultFreq("Actual") = "YTD" Then
```

```
...
```

```
End If
```

# DefaultParent

Gets the default parent for the current member or a specified member. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

## Syntax

```
HS.<Object>.DefaultParent ("Member")
```

```
HS.<Object>.DefaultParent ("")
```

```
HS.Custom("Label").DefaultParent ("Member")
```

```
HS.Custom("Label").DefaultParent ("")
```

where *Member* is a valid dimension member.



### Note:

Use a blank string (" ") to apply this function to the current member.

**Table 11-15 Syntax for DefaultParent Function**

Parameter	Description
<i>&lt;Object&gt;</i>	One of these object keywords: <ul style="list-style-type: none"> <li>• Account</li> <li>• Custom1...4</li> <li>• Custom(<i>Custom Dimension Label</i>)</li> <li>• Entity</li> <li>• Scenario</li> </ul>
<i>Member</i>	Depending on the object selected, the name of a valid member of one of these dimensions: <ul style="list-style-type: none"> <li>• Account</li> <li>• Custom1...4</li> <li>• Custom</li> <li>• Entity</li> <li>• Scenario</li> </ul>

## Return Value

A string that contains the default parent for the member.

# DefaultTranslate

Calculates translation by bypassing rules. This function overrides the Application settings and can be used in SubTranslate rules only.

## Syntax

```
HS.DefaultTranslate(dRateForBalanceAccounts, dRateForFlowAccounts,  
bUsePVAForFlowAccounts, bUsePVAForBalanceAccounts)
```

**Table 11-16 Syntax for DefaultTranslate Function**

Parameter	Description
<i>dRateForBalanceAccounts</i>	Number for rate
<i>dRateForFlowAccounts</i>	Number for rate
<i>bUserPVAForFlowAccounts</i>	True or False
<i>bUsePVAForBalanceAccounts</i>	True or False

**Return Value**

None.

**Example**

In this example, if the parent member is United States, then statements between the If...Then and End If statements are executed.

```
If HS.Parent.Member="UnitedStates" Then

    HS.DefaultTranslate .25, .27, True, False

End If
```

## DefaultView

Gets the default view for the current Scenario member or for a specified scenario member. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**

```
HS.Scenario.DefaultView("Scenario")
```

```
HS.Scenario.DefaultView("")
```

**Note:**

Use a blank string ("") to apply this function to the current member.

where *Scenario* is the name of a valid Scenario member.

### Return Value

A string that contains the default view for the specified scenario. Valid values are YTD and Periodic.

### Example

In this example, if the default view for the Actual scenario is YTD, then statements between the If...Then and End If statements are executed.

```
If HS.Scenario.DefaultView("Actual") = "YTD" Then
```

```
...
```

```
End If
```

## DefCurrency

Gets the default currency for the current entity or parent member, or for the specified entity or parent member. If you specify an entity, the system returns the entity currency. To obtain the parent currency, you must specify the parent entity. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.<Object>.DefCurrency("Entity")
```

```
HS.<Object>.DefCurrency("")
```



#### Note:

Use a blank string ("") to apply this function to the current member.

**Table 11-17** Syntax for DefCurrency Function

Parameter	Description
<Object>	One of these object keywords: <ul style="list-style-type: none"> <li>• Entity</li> <li>• Parent</li> </ul>
Entity	Name of a valid Entity member



**Return Value**

A string that contains the default currency for the specified entity or parent.

**Example**

In this example, if the default currency for Connecticut is different from the default currency for EastRegion, then statements between the If...Then and End If statements are executed.

```
If HS.Entity.DefCurrency("Connecticut") <> HS.Parent.DefCurrency("EastRegion")
Then
```

```
...
```

```
End If
```

## Down

Gets the percentage of direct ownership for the specified parent.entity node. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**

Combination of scenario, year, period, and parent.entity members.

```
HS.Node.Down("S#Scenario.Y#Year.P#Period.E#Parent.Entity")
```

```
HS.Node.Down("")
```

**Note:**

Use a blank string ("") to apply this function to the current node.

**Table 11-18 Syntax for Down Function**

Parameter	Description
<i>Scenario</i>	Name of a valid Scenario member
<i>Year</i>	A valid year
<i>Period</i>	A valid period
<i>Parent.Entity</i>	Name of a valid Parent.Entity node

### Return Value

A number that is a percentage of direct ownership.

### Example

```
dVar1 = HS.Node.Down("S#Actual.Y#2014.P#Q1.E#Group1.Ent1")
```

## Dynamic

Specifies the formula for the dynamic accounts that need calculations. This function can be used in Dynamic rules.

This function can only reference data in the same subcube. If you need to reference data from a different subcube, you may need to create an account to store information from other cubes. For example, to reference a prior year's data in the formula, you need to use an account to store the last year's data information so that it can be referenced in the dynamic calculation within the same cube.



#### Note:

You can embed the `HS.View.PeriodNumber` function in the Dynamic function. For example:

```
HS.Dynamic "A#AvgUnits = A#AccumUnits.I#ICP None.C1#None.C2#None /  
HS.View.PeriodNumber"
```

### Syntax

```
HS.Dynamic "DestPOV = Expression"
```

**Table 11-19** Syntax for Dynamic Function

Parameter	Description
<i>DestPOV</i>	A valid Account member with type set to Dynamic. You may also specify a view for which to execute the calculation.  If you do not specify a view, the formula is executed for Periodic and YTD. To have different formulas for different views, you must specify Periodic or YTD in the formula.
<i>Expression</i>	A calculation expression



#### Note:

The Dynamic function does not support IF...THEN statements.

**Return Value**

None.

**Example**

Sub Dynamic

```
HS.Dynamic "A#GM% = A#GM/(A#Sales * 100)/HS.View.PeriodNumber"
```

End Sub

Expected results for the GM% account:

Custom1	Sales	GM	GM% (Dynamic Calculation)	Calculation based on formula
Product	600	140	23.33%	140 / 600 * 100
P1	100	10	10%	10 / 100 * 100
P2	200	40	20%	40 / 200 * 100
P3	300	90	30%	90 / 300 * 100

**Exp**

Puts data into a combination of Account, ICP, and Custom1...4 members. This function can be used in these types of rules:

- Calculation
- Translation
- Allocation

**Syntax**

```
HS.Exp "DestPOV = Expression"
```

**Table 11-20 Syntax for Exp Function**

Parameter	Description
<i>DestPOV</i>	A destination point of view that identifies where to put the data. Note these usage rules: To avoid populating the database with unwanted values, rules should be as explicit as possible in terms of defining where data should reside. A good practice is to include clearly defined ICP and Custom dimension intersections for the Account dimension. Such clearly defined intersections utilize Oracle Hyperion Financial Management validation checks to avoid writing data to invalid intersections.

**Table 11-20 (Cont.) Syntax for Exp Function**

Parameter	Description
<i>Expression</i>	A calculation expression.

The destination for the data is on the left side of the equal sign, and the data to be assigned is on the right side. This example sets the cell that intersects the UnitsSold account and the [None] members of the Custom dimensions to 40000:

```
HS.Exp "A#UnitsSold.C1#[None].C2#[None].C3#[None].C4#[None]" _
    & " = 40000"
```

On the right side of the equal sign, you can use Account Expression characters to represent dimension members. Thus, you can assign data for a group of cells to another group of cells. This example sets the cell that intersects the Taxes account and the [None] members of the Custom dimensions to 6 percent of the data in the cell that intersects the Sales account and the specified Custom dimensions:

```
HS.Exp "A#Taxes.C1#[None].C2#[None].C3#[None].C4#[None]" _
    & " = A#Sales.C1#AllProducts.C2#AllCustomers.C3#[None]" _
    & ".C4#[None] * .06"
```

 **Tip:**

You can set multiple accounts with one Exp statement. See [Simultaneously Setting Several Accounts](#).

## Dimension Intersection Considerations

If you do not specify the dimension intersections from which Exp gets data and into which Exp places data, the source and destination intersections are determined by these factors:

- **Destination.** If no member of the Intercompany partner dimension or of a Custom dimension is specified on the left side of the equal sign, Exp places data into each valid intersection of the account and the dimension. If you do not specify a destination account, Oracle Hyperion Financial Management will insert data into all accounts that are valid for the current point of view. See [Simultaneously Setting Several Accounts](#).
- **Source.** If a member of a dimension is not specified on the right side of the equal sign, there are several possibilities:
  - If a dimension has only one member, then Exp gets data from the intersection of this member and the source account.

- If a dimension has only one valid intersection with the source account, then Exp gets data from this intersection.
- If a dimension has several intersecting members with the source account, then the source intersection of the data is determined by the left side of the equation:
  - \* If a member is specified on the left side, then Exp attempts to get the data from the intersection of this member and the source account.
  - \* If a member is not specified on the left side, then Exp attempts to put data into each valid intersection of the destination account and the dimension's members. Exp gets the data for the destination intersections from the corresponding intersections of the members and the source account.

 **Note:**

If a source intersection is invalid, then Exp does not change the data in the corresponding destination intersection.

For detailed examples that illustrate these considerations, see [Exp and Dimension Intersection Considerations](#).

## Period and Year Keywords

To create dynamic rules, you can use the keywords in [Table 1](#) instead of member names to represent members of the destination Year or Period dimensions:

**Table 11-21** Period and Year Keywords and Descriptions

Keyword	Description
CUR	The current period or year.
FIRST	The first period or year that is valid for the application.
LAST	The last period or year that is valid for the application.
NEXT	The period or year that follows the current period or year.
PRIOR	The period or year that precedes the current period or year.

 **Note:**

Period and Year keywords are case-sensitive and must be in all uppercase letters.

You can use the plus (+) and minus (-) signs with the Period and Year keywords. This example sets the MiscPast account to the amount in the Misc account two years before the current year.

```
HS.Exp "A#MiscPast = A#Misc.Y#CUR-2"
```

If you use the keywords Prior, First, Last, Current, or Next, immediately followed by the plus (+) and minus (-) signs and a digit, you must be careful to use the correct syntax for the order of the equation. In these cases, you can use one of these methods to write the rule:

Always use parentheses to correctly separate the variable. For example:

```
HS.Exp "A#9001_Group.C4#[None] = A#9001_Group.P#Prior" & "+" & VAR &
"*A#9001_Group.V#[ParentTotal])"
```

Or

```
HS.Exp "A#9001_Group.C4#[None] = (A#9001_Group.P#Prior" & "+" & VAR & ")
*A#9001_Group.V#[Parent Total])"
```

## Mathematical Calculations

You can add, subtract, multiply, and divide on the right side of the equal sign. You must use these standard VBScript characters:

+ - \* /

This example adds the values in the prior year's Taxes and Purchases accounts and then inserts the sum in the Cash account.

```
HS.Exp "A#Cash = A#Taxes.Y#PRIOR + A#Purchases.Y#PRIOR"
```

### Note:

If you multiply or divide with an account that has a NoData status, the data in the account on the left side of the equal sign will not be changed. Zero (0) is considered data. In other words, an account that contains 0.00 as data will *not* have a NoData status.

## Placing Other Functions Within Exp

If a function returns a single numeric value, you can nest the function within the Exp function. However, if you nest a function that contains a String argument, you cannot surround the String with quotation marks. In this example, the NumBase function is nested in the Exp function, and its String argument is *not* surrounded by quotation marks.

```
HS.Exp "A#SalesAlloc = A#Sales/HS.Entity.NumBase(Regional)"
```

## Simultaneously Setting Several Accounts

To insert data into all accounts that intersect the current point of view, use All with the account expression. You can use this to set the opening balances of all accounts. In this example, the IsFirst function tests whether the current period is the first period. If it is the first period, then

Exp sets each account's value for the current period to the account's value from the last period of the prior year.

```
If HS.Period.IsFirst = TRUE Then

    HS.Exp "A#ALL = A#ALL.Y#PRIOR.P#LAST"

End If
```

To insert data into all intersections of accounts and Custom or Intercompany Partner dimensions, use the All keyword or omit the A# characters. This example, which omits the A# characters, inserts data into each account that intersects the GolfBalls member of the Custom1 dimension. For each valid intersection of GolfBalls and an account, the amount in the prior period's intersection is placed in the current period's intersection.

```
HS.Exp "C1#GolfBalls = C1#GolfBalls.P#PRIOR"
```

### Return Value

None.

### Example

This example sets the amount in the StateTax account. The example calculates this amount by multiplying the amount in the Sales account for 2014 by the rate in the StateRate account for 2014.

```
HS.Exp "A#StateTax = A#Sales.Y#2014 * A#StateRate.Y#2014"
```

## Exp and Dimension Intersection Considerations

The following examples illustrate the considerations introduced in [Dimension Intersection Considerations](#). These types of intersections are covered:

- All intersections are valid for the source and destination accounts. See [All Intersections Valid](#).
- Some intersections are valid and others invalid for the source and destination accounts. See [Invalid Intersections](#).
- One member is valid for the source account. See [One Valid Member on the Right Side](#).

All of these examples use accounts named TargAcct and SourceAcct in conjunction with members of the Custom1 dimension named Member001, Member002, and Member003. The source intersection data for all the examples is listed in [Table 1](#):

**Table 11-22 Data for the Dimension Intersection Examples**

Member	Data in SourceAcct Intersection
Member001	10
Member002	NoData status
Member003	15

## All Intersections Valid

For this example, all intersections of the TargAcct and SourceAcct accounts and the Custom1 members are valid:

```
HS.Exp "A#TargAcct = A#SourceAcct"
```

The function puts this data into the intersections of the TargAcct account and the Custom1 members:

Custom1 Member	Data	Intersection
Member001	10	SourceAcct and Member001
Member002	---	<i>Not applicable.</i> The intersection of TargAcct and Member002 is unchanged because the intersection of SourceAcct and Member002 has a NoData status.
Member003	15	SourceAcct and Member003

This example uses Exp with the Member001 member on the left side of the equal sign:

```
HS.Exp "A#TargAcct.C1#Member001 = A#SourceAcct"
```

The intersection of TargAcct and Member001 is set to 10. Exp gets the data from the intersection of SourceAcct and Member001 because Member001 is specified on the left side.

This example uses EXP with Member003 on the right side of the equal sign:

```
HS.Exp "A#TargAcct = A#SourceAcct.C1#Member003"
```

The function puts this data into the intersections of the Custom1 members and the TargAcct account:

Custom1 Member	Data	Intersection
Member001	15	SourceAcct and Member003
Member002	15	SourceAcct and Member003
Member003	15	SourceAcct and Member003

## Invalid Intersections

In these examples, the source and destination accounts each have an invalid intersection.

- SourceAcct. Member002 and Member003 are valid, and Member001 is invalid.
- TargAcct. Member001 and Member002 are valid, and Member003 is invalid.

In this example, Exp is used without specifying a Custom1 member on either side of the equal sign:

```
HS.Exp "A#TargAcct = A#SourceAcct"
```



Exp does not change data in the TargAcct account because Exp attempts to perform these operations:

- TargAcct.Member001 = SourceAcct.Member001. SourceAcct and Member001 is an invalid intersection.
- TargAcct.Member002 = SourceAcct.Member002. Since the intersection of SourceAcct and Member002 has a NoData status, the intersection of TargAcct and Member002 remains unchanged.
- TargAcct.Member003 = SourceAcct.Member003. TargAcct and Member003 is an invalid intersection.

In this example, Exp is used with Member001 specified on the left side of the equal sign:

```
HS.Exp "A#TargAcct.C1#Member001 = A#SourceAcct"
```

TargAcct.Member001 remains unchanged because Exp attempts to retrieve data from an invalid intersection (SourceAcct and Member001).

In this example, Exp is used with Member003 specified on the right side of the equal sign:

```
HS.Exp "A#TargAcct = A#SourceAcct.C1#Member003"
```

The function puts this data into the intersections of the Custom1 members and the TargAcct account:

Custom1 Member	Data	Intersection
Member001	15	SourceAcct and Member003
Member002	15	SourceAcct and Member003
Member003	N/A	Not applicable. Member003 is an invalid intersection for the TargAcct account.

## One Valid Member on the Right Side

In these examples, the source account has only one valid member, and the destination account has two valid members.

- SourceAcct. Member003 is the only valid intersection.
- TargAcct. Member001 and Member002 are valid, and Member003 is invalid.

In this example, Exp is used without specifying a Custom1 member on either side of the equal sign:

```
HS.Exp "A#TargAcct = A#SourceAcct"
```

The function puts this data into the intersections of the Custom1 members and the TargAcct account:

Custom1 Member	Data	Intersection
Member001	15	SourceAcct and Member003 (the only valid intersection for the SourceAcct account)

Custom1 Member	Data	Intersection
Member002	15	SourceAcct and Member003 (the only valid intersection for the SourceAcct account)
Member003	N/A	<i>Not applicable.</i> Member003 is an invalid intersection for the TargAcct account.

In this example, Exp is used with Member001 specified on the left side of the equal sign:

```
HS.Exp "A#TargAcct.C1#Member001 = A#SourceAcct"
```

The intersection of TargAcct and Member001 is set to 15, which is the data in the intersection of SourceAcct and Member003.

#### Tip:

If there were more than one valid intersection for the SourceAcct account and the Custom1 dimension, Exp would attempt to get data from the intersection of SourceAcct and Member001. If this were an invalid intersection, then Exp would leave the destination account unchanged.

## GetCell

Gets the data contained in a cell. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.GetCell ("POVExpression")
```

where *POVExpression* is a valid point of view.

### Return Value

The data stored in the specified cell.

#### Note:

If the function returns more than one value, an error occurs.

## Example

This example assigns to the `dData` variable the amount stored in the intersection of the Sales account and the Golfballs member of the Custom1 dimension:

```
Dim dData

dData = HS.GetCell("A#Sales.I#[ICP
None].C1#Golfballs.C2#Customer2.C3#Increases.C4#[None]")
```

## GetCellNoData

Gets the data contained in a cell and also indicates whether the cell contains data. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.GetCellNoData("POV", Var1)
```

**Table 11-23** Syntax for GetCellNoData Function

Parameter	Description
<i>POV</i>	A valid point of view.
<i>Var1</i>	A variable that specifies if there is data in the cell.

### Return Value

The possible return values depend on what is found in the cell:

- If the cell contains real data, the data value is returned and the boolean value returned for *Var1* is False.
- If the cell contains no data, 0 is returned for the data value and the boolean value returned for *Var1* is True.
- If the cell contains derived data, the derived value is returned and the boolean value returned for *Var1* is False.

#### Caution:

If the argument causes `GetCellNoData` to return more than one value, an error occurs.

### Example

In this example, the amount in the Sales account is assigned to the dSalesAmt variable. If the Sales account has no data, the statements between the If Then and End If statements are executed.

```
dSalesAmt = HS.GetCellNoData("A#Sales.I#[ICP
None].C1#[None].C2#[None].C3#[None].C4#[None]",bIsNoData)
```

```
If bIsNoData = TRUE then
```

```
...
```

```
End If
```

## GetCellRealData

Gets the data contained in a cell and also indicates whether the cell contains real data. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.GetCellRealData("POV",Var1)
```

**Table 11-24 Syntax for GetCellRealData Function**

Parameter	Description
<i>POV</i>	A valid point of view.
<i>Var1</i>	A variable that specifies if there is real data in the cell.

### Return Value

The possible return values depend on what is found in the cell:

- If the cell contains real data, the data value is returned and the boolean value returned for *Var1* is True.
- If the cell contains no data, 0 is returned for the data value and the boolean value returned for *Var1* is False.
- If the cell contains derived data, the derived value is returned and the boolean value returned for *Var1* is False.

**Example**

```
dData = HS.GetCellRealData("A#Sales.C1#Prod1",bIsRealData)
```

```
If bIsRealData = TRUE then
```

```
...
```

```
End If
```

## GetCellText

Gets the default cell text from the point of view. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**

```
HS.GetCellText("POVExpression")
```

where *POVExpression* is a valid point of view.

**Return Value**

The [Default] label cell text for the POV.

**Example**

```
HS.GetCellText("S#Actual.Y#2014.P#January.E#Connecticut.V#<Entity  
Currency>.A#Sales.I#[ICP None].C1#[None].C2#[None].C3#[None].C4#[None]")
```

**Note:**

If you are using cell text labels, this function retrieves the cell text for the cell text label [Default].

## GetCellTextWithLabel

Gets the cell text from the Point of View for the specified cell text label. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.GetCellTextWithLabel("POVExpression", "CellTextLabel")
```

where POVExpression is a combination of members. If you do not specify a dimension, these default values are used.

- Account - [none]
- ICP - [ICP None]
- Custom1...4 - [None]
- Scenario - Current Scenario member
- Entity - Current Entity member
- Value - Current Value member
- Year and Period - Current member

CellTextLabel is either the default cell text label ([Default]) or one of the valid loaded cell text labels.



#### Note:

The default dimension member values apply when you use this function in Calculation rules. When used in Allocation, Translation, or Consolidation rules, you must specify the Account, ICP, and Custom1...4 members. The Scenario, Year, Period, View, Entity, and Value default to the current members.

### Return Value

The cell text for the POV and cell text label specified.

### Example

```
HS.GetCellTextWithLabel("A#Sales.I#[ICP  
None].C1#[None].C2#[None].C3#[None].C4#[None]", "Rating")
```

## GetCellType

Gets the cell type. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Note:**

Account types use all capital letters.

**Syntax**

```
HS.GetCellType("POVExpression")
```

where *POVExpression* is a valid point of view.

**Return Value**

The type for the specified cell.

Valid types are:

- ASSET
- LIABILITY
- REVENUE
- EXPENSE
- FLOW
- BALANCE
- BALANCERECURRING
- CURRENCYRATE
- GROUPLABEL

**Example**

This example checks to see if the cell type is EXPENSE. If it is, then statements between the If...Then and End If statements are executed.

```
If HS.GetCellType("S#Actual.C4#[None]") = "EXPENSE" Then
```

```
...
```

```
End If
```

## GetCustomLabelArray

Returns a list of Custom dimension labels in an array. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

## Syntax

```
HS.GetCustomLabelArray (Names, Aliases)
```

## Return Value

Returns two arrays, one with Custom short names and one with Custom long names.

## Example

```
HS.GetCustomLabelArray (Custom1, Products)
```

# GetItem

Gets an individual record to process for consolidation. This function can be used in Consolidation rules.



### Note:

This function can only be used for 4 or fewer Custom dimensions. For more than four Custom dimensions, use [DataUnitItem](#).

## Syntax

```
DataUnit.GetItem (lItem, strAccount, strICP, strCustom1, strCustom2,  
strCustom3, strCustom4, dData)
```

**Table 11-25** Syntax for GetItem Function

Parameter	Description
<i>lItem</i>	A record number.
<i>strAccount</i>	Name of a valid Account dimension member.
<i>strICP</i>	Name of a valid ICP dimension member.
<i>strCustom1</i>	Name of a valid Custom1 dimension member.
<i>strCustom2</i>	Name of a valid Custom2 dimension member.
<i>strCustom3</i>	Name of a valid Custom3 dimension member.
<i>strCustom4</i>	Name of a valid Custom4 dimension member.
<i>dData</i>	The data in the specified cell.

## Return Value

An array containing the account, ICP, Custom1...4, data.



## Example

```
Call DataUnit.GetItem(lItem, strAccount, strICP, strCustom1, strCustom2,
strCustom3, strCustom4, dData)
```

```
If dData = 0 Then
```

```
...
```

```
End If
```

## GetItemIDs2

Using dimension IDs, gets an individual record to process for consolidation. This function can be used in Consolidation rules.



### Note:

This function can only be used for 4 or fewer Custom dimensions. For more than four Custom dimensions, use [GetItemIDs2ExtDim](#).

## Syntax

```
DataUnit.GetItemIDs2(lItem, lAccountID, lICPID, lCustom1ID, lCustom2ID,
lCustom3ID, lCustom4ID, dData)
```

**Table 11-26** Syntax for GetItemIDs2Function

Parameter	Description
<i>lItem</i>	A record number.
<i>lAccountID</i>	ID number of a valid Account dimension member.
<i>lICPID</i>	ID number of a valid ICP dimension member.
<i>lCustom1ID</i>	ID number of a valid Custom1 dimension member.
<i>lCustom2ID</i>	ID number of a valid Custom2 dimension member.
<i>lCustom3ID</i>	ID number of a valid Custom3 dimension member.
<i>lCustom4ID</i>	ID number of a valid Custom4 dimension member.
<i>dData</i>	The data in the specified cell.

## Return Value

Variables containing the account, ICP, Custom1...4, data.

**Example**

```
Call DataUnit.GetItemIDs2(lItem, lAccount, lICP, lCustom1, lCustom2,
lCustom3, lCustom4, dData)
```

```
If dData = 0 Then
```

```
...
```

```
End If
```

## GetItemIDs2ExtDim

Using dimension IDs, gets an individual record to process for consolidation. This function can be used in Consolidation rules.

**Note:**

This function is used when you have more than four Custom dimensions.

**Syntax**

```
DataUnit.GetItemIDs2ExtDim(lItem, lAccountID, lICPID, lCustomID, dData)
```

**Table 11-27 Syntax for GetItemIDs2Function**

Parameter	Description
<i>lItem</i>	A record number.
<i>lAccountID</i>	ID number of a valid Account dimension member.
<i>lICPID</i>	ID number of a valid ICP dimension member.
<i>lCustomID</i>	ID number of a valid Custom dimension member.
<i>dData</i>	The data in the specified cell.

**Return Value**

A two-column array containing the IDs for both the dimension and member.

**Example**

```
Call DataUnit.GetItemIDs2(lItem, lAccount, lICP, lCustom5, dData)
```

```
If dData = 0 Then
```

```
...
```

```
End If
```

## GetNumItems

Gets the number of records to process for consolidation. This function can be used in Consolidation rules.

**Syntax**

```
NumItems = DataUnit.GetNumItems
```

**Return Value**

The number of records in the data unit.

**Example**

```
Set dataUnit = HS.OpenDataUnit("")
```

```
lNumItems = dataUnit.GetNumItems
```

```
for lItem = 0 to lNumItems - 1
```

```
' Get the next item from the Data Unit
```

```
Call dataUnit.GetItem(lItem, strAccount, strICP, strCustom1, strCustom2,  
strCustom3, strCustom4, dData)
```

## GetNumLID

Gets the number of line item details for the specified point of view. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation

- Allocation

### Syntax

```
HS.GetNumLID("POVExpression")
```

where *POVExpression* is a valid POV combination. If you do not specify a dimension, these values are used:

- Account - [none]
- ICP - [ICP None]
- Custom1...4 - [None]
- Scenario - Current Scenario member
- Entity - Current Entity member
- Value - <entity currency>
- Year and Period - Current member

#### Note:

If an invalid intersection is specified, the return value is 0.

### Return Value

The number of line item details entered for the specified cell.

### Example

In this example, if no line item details have been entered for the specified cell, then statements between the If...Then and End If lines are executed.

```
If HS.GetNumLID("A#Sales.C1#Prod1.C2#Region1") = 0 then
```

```
...
```

```
End If
```

## GetRate

Gets the currency rate for the current point of view or for a specified point of view. This function can be used in the these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.GetRate("POVExpression")
```

where POVExpression is a valid point of view.



#### Note:

You must specify a rate account.

### Return Value

The currency rate for the specified point of view.

### Example

```
dVar1 = HS.GetRate("S#Actual.Y#2014.P#March.V#Euro.E#Connecticut.A#AvgRate")
```

## GetSubmissionGroup

Gets the process management submission group for the cell.

### Syntax

```
HS.GetSubmissionGroup("A#Account.C1#Custom1.C2#Custom2.C3#Custom3.C4#Custom4.I  
#ICP")
```

### Return Value

An integer representing the process management submission group. Valid values are 1–99.

### Example

```
dVar1=HS.GetSubmissionGroup("A#Sales.C1#Golfballs.C2#Tennisballs.C3#Soccerball  
s.C4#Basketballs.I#EastSales")
```

## GetSubmissionPhase

Gets the process management submission phase for the cell.

### Syntax

```
HS.GetSubmissionPhase("S#Scenario.P#Period.A#Account.C1#Custom1.C2#Custom2.C3#  
Custom3.C4#Custom4.I#ICP")
```

### Return Value

An integer representing the process management submission phase. Valid values are 1–9.

### Example

```
dVar1=HS.GetSubmissionPhase("S#Actual.P#January.A#Sales.C1#Golfballs.C2#Tennis  
balls.C3#Soccerballs.C4#Basketballs.I#EastSales")
```

## Holding

Gets the holding company for the current or specified Entity or parent dimension member. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.Entity.Holding("Entity")
```

```
HS.Entity.Holding("")
```

```
HS.Parent.Holding("Entity")
```

```
HS.Parent.Holding("")
```

where *Entity* is the name of a valid Entity dimension or parent member. You can use a blank string (") to apply this function to the current member.

### Return Value

A string that contains the name of the holding company for the specified entity member or parent.

### Example

In this example, if Europe is the holding company for the entity France, then statements between the If...Then and End If lines are executed.

```
If HS.Entity.Holding("France") = "Europe" Then
```

```
...
```

```
End If
```

## ICPTopMember

Gets the ICPTopMember of the current Account dimension member or the specified account member. This function can be used in the these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.Account.ICPTopMember ("AccountName")
```

```
HS.Account.ICPTopMember ("")
```

```
HS.Account.ICPTopMember (Var1)
```



#### Note:

You can use a blank string ("") to apply this function to the current account only if you are using it in the Sub Consolidate subroutine.

**Table 11-28 Syntax for ICPTopMember Function**

Parameter	Description
<i>AccountName</i>	Name of a valid Account dimension member.
<i>Var1</i>	A VisualBasic variable.

### Return Value

A string with the name of the ICPTopMember for the account.

### Example

In this example, if the ICPTopMember of the Sales account is TotalProd, then statements between the If...Then and End If statements are executed.

```
If HS.Account.ICPTopMember("Sales") = "TotalProd" Then
```

```
...
```

```
End If
```

## ICPWeight

Gets the ICP weight for the application. The percentage of intercompany partner entity [ICP Entities] amounts that aggregate to the [ICP Top] member of the Value dimension. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.AppSettings.ICPWeight
```

### Return Value

The percentage of ICP entities that are aggregated into the ICPTopMember. The value is a percentage scaled to hundreds, with 1.0 equalling 100 percent.

### Example

In this example, if the ICPWeight of the current application is 1, then statements between the If...Then and End If lines are executed.

```
If HS.AppSettings.ICPWeight = 1 Then  
  
    ...  
  
End If
```

## IDFromMember

Gets the ID number for the specified member. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.<Object>.IDFromMember ("Element")
```



**Table 11-29 Syntax for IDFromMember Function**

Parameter	Description
<Object>	One of these object keywords: <ul style="list-style-type: none"> <li>• Account</li> <li>• Custom1...4</li> <li>• Custom (<i>Custom Dimension Label</i>)</li> <li>• Entity</li> <li>• ICP</li> <li>• Parent</li> <li>• Period</li> <li>• Scenario</li> <li>• Value</li> <li>• Year</li> <li>• View</li> </ul>
<i>Element</i>	Depending on the object selected, the name of a valid member of one of these dimensions: <ul style="list-style-type: none"> <li>• Account</li> <li>• Custom1...4</li> <li>• Custom</li> <li>• Entity</li> <li>• ICP</li> <li>• Parent</li> <li>• Period</li> <li>• Scenario</li> <li>• Value</li> <li>• Year</li> </ul>

**Return Value**

The ID number of the specified member. If the member entry is not a valid member, the return value will be -1.

**Example**

This example gets the ID number for Connecticut:

```
1EntityID = HS.Entity.IDFromMember("Connecticut")
```

```
1CustomID=HS.Custom("Prod").IDFromMember("P3000-Phones")
```

## ImpactStatus

Changes the status of the specified data unit to impacted. This function can be used in Calculation rules.

## Syntax

Combination of Scenario, Year, Period, Entity, and Value members. If the scenario is the same, the year and period combination must be a future period. If no value member is specified, it is assumed to be current.

```
HS.ImpactStatus"S#Scenario.Y#Year.P#Period.E#Entity.V#Value"
```

**Table 11-30 Syntax for ImpactStatus Function**

Parameter	Description
<i>Scenario</i>	Name of a valid Scenario dimension member.
<i>Year</i>	A valid year.
<i>Period</i>	A valid period.
<i>Entity</i>	Name of a valid Entity dimension member.
<i>Value</i>	Name of a valid Value dimension member.



### Note:

If the specified POV is the same scenario, year, period, and entity as the data unit being processed, an error occurs and there is no impact to the data unit. If the target Period is locked when the rule is run, the system returns a VBScript error and there is no impact to the data unit.

## Return Value

None.

## Example

```
HS.ImpactStatus "S#Actual.Y#2014.P#January"
```

# Input

Enables data input into <Entity Currency> of parent entities for data-points that can entered to at a base entity level. This does not include data-points flagged as IsCalculated and data-points set as NoInput. Only the entity currency Value dimension is supported. When you use this function, the value at the Parent Entity level is not equal to the sum of its children's contributions.

Because this function enables input at the Parent entity level, the children contributions do not roll up to the Parent entity's entity currency Value dimension. However, the parent-child contribution value is stored, and you can still post journals to the Contribution Adjustments Value dimension. This function can be used in Input rules.

**▲ Caution:**

For accounts where you allow input at the parent entity level, it is important to remember that the value at the Parent entity's entity currency Value member will not be equal to the sum of all the children's contributions.

**Syntax**

```
HS.Input"POVExpression"
```

where *POVExpression* is a point of view.

**Return Value**

None.

**Example**

This example enables input into the cells that intersect the Sales account and the Budget scenario:

```
Sub Input
```

```
HS.Input"A#Sales.S#Budget"
```

```
End Sub
```

## IsAlmostEqual

Checks to see if the passed in values are equal based on a predefined Oracle Hyperion Financial Management epsilon. This function can be used in all types of rules.

A difference of -0.0000000000001 to 0.0000000000001 is considered zero difference.

**Syntax**

```
BooleanValue = HS.IsAlmostEqual(Value1, Value2)
```

**Return Value**

A Boolean expression that is True if the passed in values are equal ; False if they are not equal.

**Example**

```
Dim BoolVal

Dim Value1

Dim Value2

Value1 = 10.1299999999

Value2 = 10.13

BoolVal = HS.IsAlmostEqual(Value1, Value2)

If BoolVal = true Then

    'do processing

Else

    'do Processing

End If
```

## IsBase

Determines if the current member or a specified member is a base member of the application or of the specified parent. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

 **Note:**

A member is a base member if it has no children (that is, it is at the end of branch in a tree hierarchy).

## Syntax

```
HS.<Object>.IsBase("Parent","Element")
```

```
HS.Node.IsBase("Parent"."Entity"."S#Scenario.Y#Year.P#Period")
```

```
HS.<Object>.IsBase("", "")
```

```
HS.Custom(Dimension).IsBase(Member)
```



### Note:

Use a blank string ("" ) to apply this function to the current member.

**Table 11-31** Syntax for IsBase Function

Parameter	Description
<Object>	One of these object keywords: <ul style="list-style-type: none"> <li>Account</li> <li>Custom1...4</li> <li>Custom (<i>Custom Dimension Label</i>)</li> <li>Entity</li> <li>Parent</li> </ul>
Parent	A valid Parent member. Parent is mandatory only when used with Node.
Element	Depending on the object selected, the name of a valid member of one of these dimensions: <ul style="list-style-type: none"> <li>Account</li> <li>Custom1...4</li> <li>Custom</li> <li>Entity</li> <li>Parent</li> </ul>
Entity	Name of a valid Entity dimension member.
Scenario	Name of a valid Scenario dimension member.
Year	A valid year.
Period	A valid period.



### Note:

For Node, determines if the element is an active base member of the specified parent.

### Return Value

A Boolean expression that is True if the element is a base member below the specified parent or, when no parent is specified, is a base member in the application. False if the element is not a base member.

For Node, True if the element is an active base entity below the parent in the specified point of view. False if the element is not an active base entity.

### Example

In this example, if Connecticut is a base entity under EastRegion, then statements between the If...Then and End If lines are executed.

```
If HS.Entity.IsBase("EastRegion","Connecticut") = TRUE Then
```

```
...
```

```
End If
```

```
If HS.Custom("Prod").IsBase("P3000-Phones") = TRUE Then
```

```
...
```

```
End If
```

## IsCalculated

Determines if the current Account dimension member or a specified account member is a calculated account. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.Account.IsCalculated("Account")
```

where *Account* is the name of a valid Account member.

```
HS.Account.IsConsolidated("")
```

 **Note:**

You can use a blank string ("") to apply this function to the current member only if you are using it in the Sub Consolidate subroutine.

**Return Value**

A Boolean expression that is True if the account is a calculated account; False if the account is not a calculated account.

**Example**

In this example, if the Sales account is calculated, then statements between the If...Then and End If statements are executed.

```
If HS.Account.IsCalculated("Sales") = TRUE Then
```

```
...
```

```
End If
```

## IsChild

Determines if the current member or a specified member is a child of the specified parent. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

 **Note:**

A member is a child if it is one level directly below a member in a tree hierarchy.

**Syntax**

```
HS.<Object>.IsChild("Parent", "Element")
```

```
HS.Node.IsChild("Parent"."Entity"."S#Scenario.Y#Year.P#Period")
```

```
HS.<Object>.IsChild("Parent", "")
```

```
HS.Custom(Dimension).IsChild("P3000-Phones")
```

**Note:**

Use a blank string ("") to apply this function to the current member.

**Table 11-32 Syntax for IsChild Function**

Parameter	Description
<Object>	One of these object keywords: <ul style="list-style-type: none"> <li>• Account</li> <li>• Custom1...4</li> <li>• Custom (<i>Custom Dimension Label</i>)</li> <li>• Entity</li> <li>• Parent</li> </ul>
<i>Parent</i>	Name of a Parent member. (Parent is mandatory.)
<i>Element</i>	Depending on the object selected, the name of a valid member of one of these dimensions: <ul style="list-style-type: none"> <li>• Account</li> <li>• Custom1...4</li> <li>• Custom</li> <li>• Entity</li> <li>• Parent</li> </ul>
<i>Entity</i>	Name of a valid Entity dimension member.
<i>Scenario</i>	Name of a valid Scenario dimension member.
<i>Year</i>	A valid year
<i>Period</i>	A valid period

**Note:**

For Node, determines if the member is an active child of the specified parent.

**Return Value**

A Boolean expression that is True if the element is a child of the specified parent; False if the element is not a child of the specified parent.

For Node, True if the element is an active child of the specified parent; False if the element is not an active child of the specified parent.



### Example

In this example, if Connecticut is a child of EastRegion, then statements between the If...Then and End If lines are executed.

```
If HS.Entity.IsChild("EastRegion","Connecticut") = TRUE Then
```

```
...
```

```
End If
```

```
If HS.Custom("Prod").IsChild("P3000-Phones") = TRUE Then
```

```
...
```

```
End If
```

## IsConsolidated

Determines if the current Account dimension member or a specified account member is a consolidated account. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.Account.IsConsolidated("Account")
```

where *Account* is the name of a valid Account member.

```
HS.Account.IsConsolidated("")
```

#### Note:

You can use a blank string ("") to apply this function to the current member only if you are using it in the Sub Consolidate subroutine.

### Return Value

A Boolean expression that is True if the account is consolidated into a parent account; False if the account is not consolidated into a parent account.

### Example

In this example, if the Sales account is consolidated, then statements between the If...Then and End If statements are executed.

```
If HS.Account.IsConsolidated("Sales") = TRUE Then
```

```
...
```

```
End If
```

## IsDescendant

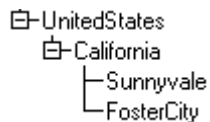
Determines if the current member or a specified member is a descendant of the specified parent. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Note:

A member is a descendant if it is at a level below a parent in a tree hierarchy. Descendants are within the same branch of the tree.

For example, in this hierarchy, FosterCity and Sunnyvale are descendants of California and UnitedStates.



### Syntax

```
HS.<Object>.IsDescendant ("Parent", "Element")
```

```
HS.Node.IsDescendant ("Parent"."Entity", "S#Scenario.Y#Year.P#Period")
```

```
HS.<Object>.IsDescendant ("Parent", "")
```

```
HS.Custom (Dimension) .IsDescendant (Member)
```

**Note:**

Use a blank string ("") to apply this function to the current member.

**Table 11-33 Syntax for IsDescendant Function**

Parameter	Description
<Object>	One of these object keywords: <ul style="list-style-type: none"> <li>• Account</li> <li>• Custom1...4</li> <li>• Custom (<i>Custom Dimension Label</i>)</li> <li>• Entity</li> <li>• Parent</li> </ul>
<i>Parent</i>	Name of a valid Parent member. Parent is required.
<i>Element</i>	Depending on the object selected, name of a valid member of one of these dimensions: <ul style="list-style-type: none"> <li>• Account</li> <li>• Custom1...4</li> <li>• Custom</li> <li>• Entity</li> <li>• Parent</li> </ul>
<i>Entity</i>	Name of a valid Entity dimension member.
<i>Scenario</i>	Name of a valid Scenario dimension member.
<i>Year</i>	A valid year.
<i>Period</i>	A valid period.

**Note:**

When you use node as the object, the function determines if the member is an active descendant of the specified parent.

**Return Value**

A Boolean expression that is True if the element is a descendant of the specified parent; False if the element is not a descendant of the specified parent.

For Node, True if the element is an active descendant of the specified parent; False if the element is not an active descendant of the specified parent.

### Example

In this example, if Connecticut is a descendant of Regional, then statements between the If...Then and End If lines are executed.

```
If HS.Entity.IsDescendant("Regional","Connecticut") = TRUE Then
```

```
...
```

```
End If
```

```
If HS.Custom("Prod").IsDescendant("All_Phones", P3000_Phones")
```

```
...
```

```
End If
```

## IsFirst

Determines if the current period or year is the first period or year of the application. The default frequency of the current scenario is used to determine if the current period or year is the first period or year of the application. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.<Object>.IsFirst
```

where <Object> is one of these keywords:

- Period
- Year

### Return Value

A Boolean expression that is True if the current period or year is the first period or year; False if the current period or year is not the first period or year.

### Example

In this example, if the current period is the first period then statements between the If...Then and End If statements are executed:

```
If HS.Period.IsFirst = TRUE Then
```

```
...
```

```
End If
```

## IsICP

Determines if the current Account or Entity dimension member or a specified account or entity member is an intercompany partner (ICP). This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.<Object>.IsICP("Element")
```

```
HS.<Object>.IsICP("")
```



#### Note:

Use a blank string ("") to apply this function to the current member.

**Table 11-34** Syntax for IsICP Function

Parameter	Description
<Object>	One of these object keywords: <ul style="list-style-type: none"> <li>• Account</li> <li>• Entity</li> </ul>
<i>Element</i>	Depending on the object selected, name of a valid member of the Account or Entity dimension. <ul style="list-style-type: none"> <li>• Account</li> <li>• Entity</li> </ul>

### Return Value

A Boolean expression that is True if the account or entity member is an intercompany partner; False if the account or entity member is not an intercompany partner.

### Example

In this example, if the Sales account is an intercompany partner, then statements between the If...Then and End If lines are executed.

```
If HS.Account.IsICP("Sales") = TRUE Then
```

```
...
```

```
End If
```

## IsLast

Determines if the current period or year is the last period or year of the application. The default frequency of the current scenario is used to determine if the current period or year is the last period or year of the application. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.<Object>.IsLast
```

where <Object> is one of these keywords:

- Period
- Year

### Return Value

A Boolean expression that is True if the current period or year is the last period or year; False if the current period or year is not the last period or year.

**Example**

In this example, if the current period is the last period, then statements between the If...Then and End If statements are executed:

```
If HS.Period.IsLast = TRUE Then  
  
    ...  
  
End If
```

## IsTransCur

Determines if the current Value dimension member is a translated currency member. This function can be used in Calculation rules.

**Syntax**

```
HS.Value.IsTransCur
```

**Return Value**

A Boolean expression that is True if the current Value member is a translated currency member; False if the current Value member is not a translated currency member.

**Example**

In this example, if the Value member is a translated currency member, then all statements between the If...Then and End If statements are executed.

```
If HS.Value.IsTransCur = TRUE Then  
  
    ...  
  
End If
```

## IsTransCurAdj

Determines if the current Value dimension member is a translated currency Adj member. This function can be used in Calculation rules.

**Syntax**

```
HS.Value.IsTransCurAdj
```

### Return Value

A Boolean that is True if the current Value member is a translated currency Adj member; False if the current Value member is not a translated currency Adj member.

### Example

In this example, if the Value member is a translated currency Adj member, then all statements between the If...Then and End If statements are executed.

```
If HS.Value.IsTransCurAdj = TRUE Then
```

```
...
```

```
End If
```

## IsValidDest

Determines if the specified point of view is a valid destination. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Note:

This function does not check to see whether the cell is a calculated cell.

### Syntax

```
HS.IsValidDest ("POVExpression")
```

where *POVExpression* is a point of view. If you do not specify a dimension, these default values are used:

- Account - Current Account member if used in Sub Consolidate subroutine. Otherwise, account is required.
- Custom and ICP - Current member is used in Sub Consolidate subroutine. Otherwise, the TopMember for the account is used.
- Scenario - Current Scenario member
- Entity - Current Entity member
- Value - Current Value member
- Year and Period - Current member



### Return Value

A Boolean that is True if the specified point of view is a valid destination; False otherwise.

### Example

In this example, if the specified destination is valid, then all statements between the If...Then and End If statements are executed.

```
If HS.IsValidDest("A#Sales.I#CT.C1#P1.C2#R1.C3#[None].C4#[None]") = TRUE Then  
  
    ...  
  
End If
```

## IsZero

Checks to see if the passed in value is close to zero based on a predefined Oracle Hyperion Financial Management epsilon. This function can be used in all types of rules.

This function is recommended instead of an exact comparison to zero where floating point arithmetic introduces errors of less than  $1 \times 10^{-10}$  that can be ignored.

Instead of:

```
Difference = Value1 - Value2  
  
If Difference = 0 Then  
  
    'process where Difference = 0  
  
Else  
  
    'process where Difference <> 0  
  
End If
```

**Use:**

```
Difference = Value1 - Value2
```

```
If HS.IsZero(Difference) Then
```

```
    'process where Difference = 0
```

```
Else
```

```
    'process where Difference <> 0
```

```
End If
```

**Syntax**

```
BooleanValue = HS.IsZero(Value)
```

**Return Value**

A Boolean that is True if the passed in value is close to zero. False otherwise.

**Example**

```
Dim BoolVal
```

```
Dim Value
```

```
Value = 0.000000001
```

```
BoolVal = HS.IsZero(Value)
```

```
If BoolVal = true Then
```

```
    'do processing
```

```
Else
```

```
    'do Processing
```

```
End If
```

# List

Gets the elements contained in the specified list. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

## Syntax

```
HS.<Object>.List("Parent","Listname")
```

```
HS.Node.List("Parent","Listname"."S#Scenario.Y#Year.P#Period")
```

```
HS.Custom("Dimension").List("Parent","Listname")
```

**Table 11-35 Syntax for List Function**

Parameter	Description
<Object>	One of these object keywords: <ul style="list-style-type: none"> <li>• Account</li> <li>• Custom1...4</li> <li>• Custom (<i>Custom Dimension Label</i>)</li> <li>• Entity</li> <li>• Parent</li> <li>• ICP</li> <li>• Scenario</li> </ul>
<i>Parent</i>	Name of a valid Parent member.
<i>Listname</i>	Name of a valid system list or user-defined list. For node, must be a valid entity system list.
<i>Scenario</i>	Name of a valid Scenario dimension member.
<i>Year</i>	A valid year.
<i>Period</i>	A valid period.

## Return Value

An array that contains all elements of the specified list. For Node, only the active elements in the list.

## Example

This example gets the elements of the user-defined list MyBaseList for the current account:

```
HS.Account.List("", "MyBaseList")
```

This example gets the elements of the system list [Base] for the TotalAssets account:

```
HS.Account.List("TotalAssets", "[Base]")
```

```
Hs.Custom("Prod").List("Products", "[Base]")
```

## Member

Gets the name of the current member. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.<Object>.Member
```

where <Object> is one of these keywords:

- Entity
- Parent
- Period
- Scenario
- Value
- Year
- View

### Return Value

A string that contains the current member name.

For the Value object, Member returns the name of the current Value member, not the currency associated with the value. For example, if the current Value member is Entity Currency and the value is associated with the USD currency, HS.Parent.Member returns Entity Currency, not USD.



#### Tip:

To get the currency of the current Value member, use the DefCurrency function.

**Example**

In this example, if the current entity is California then statements between the If...Then and End If statements are executed:

```
If HS.Entity.Member = "California" Then
```

```
...
```

```
End If
```

## MemberFromID

Gets the dimension member for the specified ID number. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**

```
HS.<Object>.MemberFromID(ElementID)
```

```
HS.Custom(Dimension).MemberFromID(ElementID)
```

**Table 11-36 Syntax for MemberFromID Function**

Parameter	Description
<Object>	One of these object keywords: <ul style="list-style-type: none"> <li>• Account</li> <li>• Custom1...4</li> <li>• Custom (<i>Custom Dimension Label</i>)</li> <li>• Entity</li> <li>• ICP</li> <li>• Parent</li> <li>• Period</li> <li>• Scenario</li> <li>• Value</li> <li>• Year</li> <li>• View</li> </ul>
<i>Parent</i>	A valid Parent member.

**Table 11-36 (Cont.) Syntax for MemberFromID Function**

Parameter	Description
<i>ElementID</i>	Depending on the object selected, the ID number of a valid member of one of these dimensions: <ul style="list-style-type: none"> <li>• Account</li> <li>• Custom1...4</li> <li>• Custom</li> <li>• Entity</li> <li>• ICP</li> <li>• Parent</li> <li>• Period</li> <li>• Scenario</li> <li>• Value</li> <li>• Year</li> </ul>

**Return Value**

The dimension member name.

**Example**

This example gets the member for the ID number 001:

```
strEntity = HS.Entity.MemberFromID(001)
```

```
strCustom = HS.Custom("Prod").MemberFromID(001)
```

## Method

Gets the consolidation method for the specified member. If there is more than one non-zero value, the system returns the first one found. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**

```
HS.Node.Method("POVExpression")
```

where *POVExpression* is a combination of Scenario, Year, Period, and Entity members.

**Return Value**

A string that specifies the consolidation method for the specified point of view.

### Example

In this example, if the method for the point of view is GLOBAL, then statements between the If...Then and End If statements are executed.

```
If HS.Node.Method("S#Actual.Y#2014.P#January.E#Regional.Connecticut") =  
"GLOBAL" Then  
  
    ...  
  
End If
```

## NoInput

Prevents users from entering data into specific cells or slices of cells. This is useful when there are accounts that are not input or calculated.

When you set up a calculated account, you in effect, are preventing users from being able to input to that account. However, if you have accounts in which data input is enabled for some dimension intersections but not for others, you can use NoInput. This function can be used in NoInput rules.

### Syntax

```
HS.NoInput "POVExpression"
```

where *POVExpression* is a point of view.

### Return Value

None.

### Example

This example prohibits input into the cells that intersect the Sales account and the Budget scenario for 2014:

```
Sub NoInput  
  
    HS.NoInput "S#Budget.Y#2014.A#Sales"  
  
End Sub
```

## NoRound

Turns off rounding for all following `Exp` statements. This function can be used in these types of rules:

- Calculation

- Translation
- Allocation

 **Tip:**

You can also turn off rounding by entering 0 as the argument for the Round function. For example, HS.Round(0) turns off rounding.

**Syntax**

```
HS.NoRound
```

**Return Value**

None.

**Example**

This example rounds the amount inserted into the SalesRound account's cells to the nearest tenth, then uses NoRound to turn off rounding for the amount inserted into the SalesNoRound account's cells:

```
HS.Round 0.1
```

```
HS.Exp "A#SalesRound" = "A#Sales"
```

```
HS.NoRound
```

```
HS.Exp "A#SalesNoRound" = "A#Sales"
```

## NumBase

Gets the number of base members for the current member or for a specified member. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

 **Note:**

A member is a base member if it has no children (that is, it is at the end of branch in a tree hierarchy).



## Syntax

```
HS.<Object>.NumBase("Element")
```

```
HS.Node.NumBase("S#Scenario.Y#Year.P#Period.E#Entity")
```

```
HS.<Object>.NumBase("")
```

```
HS.Custom(Dimension).NumBase(Member)
```



### Note:

Use a blank string (") to apply this function to the current member.

**Table 11-37 Syntax for NumBase Function**

Parameter	Description
<Object>	One of these object keywords: <ul style="list-style-type: none"> <li>Account</li> <li>Custom1...4</li> <li>Custom (<i>Custom Dimension Label</i>)</li> <li>Entity</li> <li>Parent</li> <li>Node</li> </ul>
Element	Depending on the object selected, name of a valid member for one of these dimensions: <ul style="list-style-type: none"> <li>Account</li> <li>Custom1...4</li> <li>Custom</li> <li>Entity</li> <li>Parent</li> <li>Node</li> </ul> <p>For Account and Custom objects, you must specify the member - you cannot use a blank string.</p> <p>To get the number of base members in the entire dimension, specify ALL within quotation marks, as in this example:</p> <pre>iAcctBase = HS.Account.NumBase("ALL")</pre>
Scenario	Name of a valid Scenario dimension member.
Year	A valid year.
Period	A valid period.
Entity	Name of a valid Entity dimension member.

You can also embed the `NumBase` function in the `Exp` function. If you embed the `NumBase` function, do not surround `NumBase`'s argument with quotation marks, as in this example:

```
HS.Exp "A#AverageSales = A#Sales/HS.Entity.NumBase(Regional) "
```

### Return Value

A Long that identifies the number of base members. For Node, gets the number of active base elements of the specified member.



#### Note:

If a base entity appears twice in a branch, the entity is counted twice.

### Example

In this example, the application contains an account named `SalesAlloc` that stores the average sales amount for the base entities under the `Regional` entity. To calculate the `SalesAlloc` amount, the example divides the `Sales` account's amount by the number of base entities under `Regional`.

```
If HS.Exp"A#SalesAlloc = A#Sales/HS.Entity.NumBase(Regional) " then
```

```
...
```

```
End If
```

```
If HS.Exp"ASalesAlloc = A#Sales/HS.Custom("Prod").NumBase("TotalProducts")
```

```
...
```

```
End If
```

## Number

Gets the current period number. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.View.PeriodNumber
```

### Return Value

The current period number.

### Example

In this example, if the current period is the first period then statements between the If...Then and End If statements are executed.

```
If HS.View.PeriodNumber = 1 Then
```

```
...
```

```
End If
```

## NumChild

Gets the number of child members for the current dimension member or for a specified member. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

 **Note:**

A member is a child if it is one level directly below a member in a tree hierarchy. Only members one level below the specified object are counted.

### Syntax

```
HS.<Object>.NumChild("Element")
```

```
HS.Node.NumChild("S#Scenario.Y#Year.P#Period.E#Entity")
```

```
HS.<Object>.NumChild("")
```

```
HS.Custom(Dimension).NumChild(Member)
```

 **Note:**

Use a blank string ("" ) to apply this function to the current member.

**Table 11-38 Syntax for NumChild Function**

Parameter	Description
<i>&lt;Object&gt;</i>	One of these object keywords: <ul style="list-style-type: none"> <li>Account</li> <li>Custom1-4</li> <li>Custom (<i>Custom Dimension Label</i>)</li> <li>Entity</li> <li>Node</li> <li>Parent</li> </ul>
<i>Element</i>	Depending on the object selected, name of a valid member of one of these dimensions: <ul style="list-style-type: none"> <li>Account</li> <li>Custom1...4</li> <li>Custom</li> <li>Entity</li> <li>Node</li> <li>Parent</li> </ul> <p>For Account and Custom objects, you must specify the member - you cannot use a blank string.</p> <p>To get the number of child members in the entire dimension, specify ALL within quotation marks, as in this example:</p> <pre>iAcctBase = HS.Account.NumChild("ALL")</pre>
<i>Scenario</i>	Name of a valid Scenario dimension member.
<i>Year</i>	A valid year.
<i>Period</i>	A valid period.
<i>Entity</i>	Name of a valid Entity dimension member.

You can embed the NumChild function in the Exp function. If you embed the NumChild function, do not surround NumChild's argument with quotation marks.

### Return Value

A string that identifies the number of child members. For Node, gets the number of active children of the specified member.

### Example

In this example, the application contains an account named SalesChild that stores the average sales amount for the entities immediately under the Regional entity. To calculate the SalesChild amount, the example divides the Sales account's amount by the number of children directly under Regional.

```
HS.Exp "A#SalesChild = A#Sales/HS.Entity.NumChild(Regional)"
```

## NumCustom

Returns the total number of Custom dimensions defined for the application. For example, if you create five Custom dimensions, delete one Custom dimension, and add two Custom dimensions, the value returned for this function should be six, which is the total number of

Custom dimensions defined for the application. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.NumCustom
```

### Return Value

The number of Custom dimensions for the application.

### Example

```
nCustoms = HS.NumCustom
```

## NumDescendant

Gets the number of descendants of the current dimension member or a specified member. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Note:

A member is a descendant if it is at a level below a parent in a tree hierarchy. Descendants are within the same branch of the tree.

For example, in this hierarchy, FosterCity and Sunnyvale are descendants of California and UnitedStates.



## Syntax

```
HS.<Object>.NumDescendant ("Element")
```

```
HS.Node.NumDescendant ("S#Scenario.Y#Year.P#Period.E#Entity")
```

```
HS.<Object>.NumDescendant ("")
```

```
HS.Custom (Dimension) .NumDescendant (Member)
```



### Note:

Use a blank string "" to apply this function to the current member.

**Table 11-39 Syntax for NumDescendant Function**

Parameter	Description
<Object>	<p>One of these object keywords:</p> <ul style="list-style-type: none"> <li>Account</li> <li>Custom1...4</li> <li>Custom (<i>Custom Dimension Label</i>)</li> <li>Entity</li> <li>Node</li> <li>Parent</li> </ul>
Element	<p>Depending on the object selected, name of a valid member of one of these dimensions:</p> <ul style="list-style-type: none"> <li>Account</li> <li>Custom1...4</li> <li>Custom</li> <li>Entity</li> <li>Node</li> <li>Parent</li> </ul> <p>For Account and Custom objects, you must specify the member - you cannot use a blank string.</p> <p>To get the number of descendant in the entire dimension, specify ALL within quotation marks, as the argument as in this example:</p> <pre>iAcctBase = HS.Account.NumDescendant ("ALL")</pre>
Scenario	Name of a valid Scenario dimension member.
Year	A valid year.
Period	A valid period.
Entity	Name of a valid Entity dimension member.

### Return Value

The number of descendants of the specified member. For node, the number of active descendant entities below the specified member.



#### Note:

If a descendant entity appears twice in a branch, the entity is counted twice.

### Example

In this example, if the entity France has no descendants, then statements between the If...Then and End If statements are executed.

```
If HS.Entity.NumDescendant("France") = 0 Then
```

```
...
```

```
End If
```

```
If HS.Custom("Product").NumDescendant("P3000-Phones") = 0 Then
```

```
...
```

```
End If
```

## NumPerInGen

Gets the number of periods in the generation for the current period being processed. This function can be used in Dynamic SUB functions.

### Syntax

```
HS.Period.NumPerInGen
```

### Return Value

One value for the number of periods of the view.

### Example

```
HS.Dynamic "A#MarginPct=A#GrossMargin/HS.Period.NumPerInGen"
```

For example, if the current period is April, and April is in the fourth generation in the calendar file (monthly generation), the number of periods for the monthly generation is 12. If the current

period is Q2, which is in the third generation of the calendar file (quarterly generation), the number of periods is 4.

Monthly generation (4th generation):

January, February, March, April, May, June, July, August, September, October, November, December

System returns 12 for the number of periods in this generation.

Quarterly generation (3rd generation):

Q1, Q2, Q3, Q4

System returns 4 for the number of periods in this generation.

Half-yearly generation (2nd generation):

HY1, HY2

System returns 2 for the number of periods in this generation.

Yearly generation (first generation):

Year

System returns 1 for the number of periods in this generation.

## NumPeriods

Gets the number of periods defined for the frequency of the specified scenario. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.Scenario.NumPeriods("ScenarioName")
```

or

```
HS.Scenario.NumPeriods(" ")
```

or

```
HS.Scenario.NumPeriods(Var1)
```

### Return Value

Numeric value for the number of periods for the frequency. For example, if the scenario is monthly, the system returns 12 for the number of periods. If the scenario is quarterly, the system returns 4 for the number of periods.



**Example**

This example returns the number of periods defined for the frequency of the Actual scenario.

```
HS.Scenario.NumPeriods("Actual")
```

## OpenCellTextUnit

Retrieves the cell text for multiple cells. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation

You can specify part of the POV for which to retrieve cell text, for example, Scenario and Year, instead of the entire POV. If you do not specify a page dimension member, (Scenario, Year, Period, Value, Entity), then cell text is returned only for the current member. If you do not specify a sub-cube dimension (Account, ICP, Custom), then the system returns cell text for all base and parent members. You can specify the Scenario and Year member to retrieve cell text for a sub-cube that is not the current sub-cube. You can also specify a member list for Period, Value, and/or Entity to retrieve cell text for more than one sub-cube. You can specify to retrieve cell text labels, and also whether to sort the cell text by Ascending or Descending order of dimension member or cell text label. If you do not specify a value for sorting, then no sorting is performed on the cell text unit, and results are returned based on their order in the database.

**Syntax**

```
Set CTU =  
HS.OpenCellTextUnit(POVExpression,Label(s),Sort_Dimension,Sort_Order)
```

Where *POVExpression* is a POV, *Label* is none, one, or more cell text labels, and *Sort\_Order* is Ascending or Descending.

**Return Value**

Returns information on all the cell text and labels for the specified POV.

**Example**

```
Set CTU = HS.OpenCellTextUnit("S#Actual.Y#2014","", "", "")
```

```
Set CTU = HS.OpenCellTextUnit("S#Actual.Y#2014.P{.[Base]}.A{.  
[Base]}","", "Period", "Ascending")
```

```
Set CTU = HS.OpenCellTextUnit("S#Actual.Y#2014.E{.[Base]}.P{.  
[Base]}", "CouponRate", "LABEL", "Ascending")
```

```
Set CTU = HS.OpenCellTextUnit("S#Actual.Y#2014.E{.[Base]}.P{.  
[Base]}.A#Purchases","", "Brands", "Descending")
```

If you want to retrieve cell text for multiple labels, you can enter the labels as comma-separated values. If you do not specify any value for Label(s), all labels are retrieved.

### Example

```
Set CTU =  
HS.OpenCellTextUnit("A#Sales", "Label-1,Label-2", "Label", "Descending")
```

## OpenDataUnit

Gets the data unit to process during consolidation, calculation, or translation. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation

### Syntax

```
HS.OpenDataUnit(POVExpression)
```

where *POVExpression* is a POV. As part of the POV Expression, the function supports user-defined and system lists for Account, ICP, C1, C2, C3, and C4.

### Return Value

When used in the Sub Consolidate subroutine, returns all records with data but returns only accounts that are flagged as consolidated.

When used in the Sub Calculate or Sub Translate subroutine, returns all records containing data, including accounts that are flagged as consolidated.



#### Note:

An account is consolidated if its IsConsolidated attribute = True.

### Example

```
Set DataUnit=HS.OpenDataUnit{"A{TotalRev.[Base]}.C1{C1Top.  
[Base]}.C2{MyC2List}.C3#[None]"}
```

## OpenDataUnitSorted

Gets the data units to process during calculation, translation, or consolidation, with data sorted in the order specified. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation

## Syntax

```
HS.OpenDataUnitSorted(POVExpression, <dimension to be sorted>, Ascending or  
Descending)
```

where *POVExpression* is a POV string, <*dimension to be sorted*> is a dimension name string, and can only be one of these six dimensions:

"Account" or "A", "ICP" or "I", "Custom1" or "C1", "Custom2" or "C2", "Custom3" or "C3", or "Custom4" or "C4".

You must specify one of the following: Account, ICP, Custom1, Custom2, Custom3, Custom4.

The third parameter is a string value: "Ascending" or "A", or "Descending" or "D".

## Examples

```
Set DataUnit= HS.OpenDataUnitSorted("S#Actual.E#Group1", "C1", "Ascending")
```

# Owned

Gets the owned entity of the entity pair currently processed. This function is used in Equity Pickup rules.

## Syntax

```
HS.Entity.Owned
```

## Return Value

The owned entity.

## Example

```
Owned=HS.Entity.Owned
```

# Owner

Gets the owner of the entity pair currently processed. This function is used in Equity Pickup rules.

## Syntax

```
HS.Entity.Owner
```

## Return Value

The owner entity.

## Example

```
Owner=HS.Entity.Owner
```

## PCon

Gets the percentage of consolidation for the current member or a specified member. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.Node.PCon("S#Scenario.Y#Year.P#Period.E#Entity")
```

```
HS.Node.PCon("")
```



#### Note:

Use a blank string ("" ) to apply this function to the current member.

**Table 11-40 Syntax for PCon Function**

Parameter	Description
Scenario	Name of a valid Scenario dimension member.
Year	A valid year.
Period	A valid period.
Entity	Name of a valid Entity dimension member.

### Return Value

The percentage of consolidation for the member.

### Example

This example gets the percent consolidation for the specified point of view:

```
Pcon = HS.Node.PCon("S#Actual.Y#2014.P#January.E#Regional.Connecticut")
```

## PEPU

Gets the percentage of ownership from the Equity Pickup (EPU) table. This function is used in Equity Pickup rules.

**Syntax**

```
HS.PEPU("S#.Y#.P#", Owner, Owned)
```

**Return Value**

The ownership percentage from the EPU table.

**Example**

```
HS.PEPU(S#Actual.Y#2014.P#Jan, Group, CT)
```

or

```
""
```

```
HS.PEPU("", "", "")
```

Default parameters: if the values are blank, the function returns the percentage of ownership for the entity pair in the current Scenario, Year, and Period.

**Example**

```
Sub EquityPickUp()
```

```
Owned=Hs.Entity.Owned
```

```
OwnerCurrencyTotl=Hs.Entity.DefCurrency & "Total"
```

```
Hs.Clear "A#Inv.C4#EPU.I#" & Owned
```

```
Hs.Exp "A#Inv.C4#EPU.I#" & Owned & "=A#EQ.C4#C3Tot.I#[ICPTot].E#" &
```

```
Owned & ".V#" & OwnerCurrencyTotl & "*" & Hs.PEPU (,,)
```

```
End Sub
```

## PeriodNumber

Gets the period number in the view for the data that is being retrieved. This function can be used in Dynamic SUB functions.

**Syntax**

```
HS.View.PeriodNumber
```

## Return Value

One value for the number of periods of the view.

## Example

```
Hs.Dynamic "A#MarginPct=A#GrossMargin/HS.View.PeriodNumber"
```

Period	Periodic	YTD	QTD	HYTD
Jan.	1	1	1	1
Feb.	1	2	2	2
Mar.	1	3	3	3
Q1	1	1	1	1
Apr.	1	4	1	4
May	1	5	2	5
June	1	6	3	6
Q2	1	2	1	2
HY1	1	1	1	1
July	1	7	1	1
Aug.	1	8	2	2
Sept.	1	9	3	3
Q3	1	3	1	1
Oct.	1	10	1	4
Nov.	1	11	2	5
Dec.	1	12	3	6
Q4	1	4	1	2
HY2	1	2	2	1
Year	1	1	1	1

## PlugAcct

Gets the plug account for the current Account member or for a specified account. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.Account.PlugAcct("Account")
```

```
HS.Account.PlugAcct("")
```

where *Account* is the name of a valid Account dimension member.

**Note:**

You can use a blank string ("") to apply this function to the current member only if you are using it in the Sub Consolidate subroutine.

**Return Value**

A string that specifies the name of the plug account for the member.

**Example**

In this example, if the plug account for the Sales account is Plug1 then statements between the If...Then and End If statements are executed:

```
If HS.Account.PlugAcct("Sales") = "Plug1" Then
```

```
...
```

```
End If
```

## POwn

Gets the ultimate percentage of ownership for the current member or for a specified member. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**

```
HS.Node.POwn("S#Scenario.Y#Year.P#Period.E#Entity")
```

```
HS.Node.POwn("")
```

**Note:**

Use a blank string ("") to apply this function to the current member.

**Table 11-41 Syntax for POwn Function**

Parameter	Description
<i>Scenario</i>	Name of a valid Scenario dimension member.

**Table 11-41 (Cont.) Syntax for POwn Function**

Parameter	Description
<i>Year</i>	A valid year.
<i>Period</i>	A valid period.
<i>Entity</i>	Name of a valid Entity dimension member.

**Return Value**

The percentage of ownership for the member.

**Example**

This example gets the percent ownership for the specified point of view:

```
POwn = HS.Node.POwn("S#Actual.Y#2014.P#January. E#Regional.Connecticut")
```

## PVAForBalance

Determines the default translation method for BALANCE accounts (ASSET and LIABILITY accounts). This function overrides the application defaults for currency conversions during translation. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**

```
HS.AppSettings.PVAForBalance
```

**Return Value**

A Boolean expression that is True if BALANCE accounts use the periodic value (PVA) translation method; False if BALANCE accounts use the value at exchange rate (VAL) translation method.

**Example**

In this example, if BALANCE accounts in the application use the periodic value translation method, then statements between the If...Then and End If statements are executed:

```
If HS.AppSettings.PVAForBalance = TRUE Then
```

```
...
```

```
End If
```



## PVAForFlow

Determines the default translation method for FLOW accounts (REVENUE and EXPENSE accounts). This function overrides the application defaults for currency conversions during translation. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.AppSettings.PVAForFlow
```

### Return Value

A Boolean expression that is True if FLOW accounts use the periodic value (PVA) translation method; False if FLOW accounts use the value at exchange rate (VAL) translation method.

### Example

In this example, if FLOW accounts in the application use the value at exchange rate translation method, then statements between the If...Then and End If statements are executed:

```
If HS.AppSettings.PVAForFlow = FALSE Then
```

```
...
```

```
End If
```

## RateForBalance

Gets the default translation rate account to use for BALANCE accounts (ASSET and LIABILITY accounts). This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.AppSettings.RateForBalance
```

### Return Value

A string that specifies the rate account containing the default translation rate to use for BALANCE accounts.

### Example

In this example, if the default translation rate account for BALANCE accounts is Rate1, then statements between the If...Then and End If statements are executed:

```
If HS.AppSettings.RateForBalance = "Rate1" Then
```

```
...
```

```
End If
```

## RateForFlow

Gets the default translation rate account to use for FLOW accounts (REVENUE and EXPENSE accounts). This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.AppSettings.RateForFlow
```

### Return Value

A string that specifies the rate account containing the default translation rate to use for FLOW accounts.

### Example

In this example, if the default translation rate account for FLOW accounts is Rate2, then statements between the If and End If statements are executed:

```
If HS.AppSettings.RateForFlow = "Rate2" Then
```

```
...
```

```
End If
```

## ReviewStatus

Gets the review status for the specified point of view. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.ReviewStatus("S#Scenario.Y#Year.P#Period.E#Entity.V#Value")
```

```
HS.ReviewStatus("")
```



#### Note:

Use a blank string ("" ) to apply this function to the current member.

**Table 11-42 Syntax for ReviewStatus Function**

Parameter	Description
Scenario	Name of a valid Scenario dimension member.
Year	A valid year
Period	A valid period.
Entity	Name of a valid Entity dimension member.
Value	Name of a Value dimension member.

### Return Value

A string that specifies the review status for the member. Valid review statuses are as follows:

- Not Started
- First Pass
- Review Level 1-10
- Submitted
- Approved
- Published
- Not Supported

### Example

In this example, if the review status of the specified point of view is Submitted then statements between the If...Then and End If statements are executed:

```
If HS.ReviewStatus("") = "Submitted" Then  
  
...  
  
End If
```

## ReviewStatusUsingPhaseID

Gets the review status for the specified point of view using the process management submission phase ID.

### Syntax

```
HS.ReviewStatusUsingPhaseID("S#Scenario.Y#Year.P#Period.E#Entity",n)
```

where *n* is an integer representing the process management submission phase. Valid values are 1–9.

### Return Value

A string that specifies the review status for the member using the Submission Phase ID. Valid review statuses are as follows:

- Not Started
- First Pass
- Review Level 1-10
- Submitted
- Approved
- Published
- Not Supported

### Example

```
HS.ReviewStatusUsingPhaseID("S#Actual.Y#2014.P#January.E#Connecticut",3)
```

## Round

Rounds data from the Exp function. You specify the degree of rounding in the argument. This function can be used in these types of rules:

- Calculation
- Translation

- Consolidation

**Tip:**

If you need to apply various degrees of rounding in a Calculation rule, you can include multiple statements that contain Round.

**Syntax**

```
HS.Round(Unit)
```

where *Unit* is a factor for rounding. Value of 1 rounds to the nearest whole number. Value of 0.1 rounds to the nearest tenth. Value of 0 turns off rounding.

If you specify 0 for this argument, rounding is turned off for all subsequent Exp functions in a Calculation rule. This syntax has the same effect as HS.NoRound: HS.Round(0)

**Caution:**

The NumDecimalPlaces attribute of an account determines the maximum number of digits that can appear to the right of the decimal point. The Round function does not override this attribute.

**Return Value**

None.

**Example**

This example rounds the amount inserted into the SalesRound account to the nearest tenth, then uses NoRound to turn off rounding for the amount inserted into the SalesNoRound account's cells:

```
HS.Round(0.1)
```

```
HS.Exp"A#SalesRound = A#Sales"
```

```
HS.NoRound
```

```
HS.Exp"A#SalesNoRound = A#Sales"
```

## Scale

Gets the scale of the specified currency. This function can be used in these types of rules:

- Calculation
- Translation

- Consolidation
- Allocation

### Syntax

```
HS.Currency.Scale("Currency")
```

```
HS.Currency.Scale(Var1)
```

**Table 11-43 Syntax for Scale Function**

Parameter	Description
<i>Currency</i>	Name of a valid currency.
<i>Var1</i>	A VisualBasic variable.

### Return Value

A number indicating the scale of the specified currency (0 to 9). Specifies the unit in which amounts are displayed and stored for the currency by identifying where the decimal point is placed. The return values are as follows:

- 0 = Units
- 1 = Tens
- 2 = Hundreds
- 3 = Thousands
- 4 = Ten Thousands
- 5 = Hundred Thousands
- 6 = Millions
- 7 = Ten Millions
- 8 = Hundred Millions
- 9 = Billions

### Example

In this example, if the scale for French francs (FF) is 3, then statements between the If...Then and End If statements are executed:

```
If HS.Currency.Scale("FF") = 3 Then
```

```
...
```

```
End If
```

## SecurityAsPartner

Gets the security class assigned to the specified entity or parent when the entity or parent is used as an intercompany partner. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.Entity.SecurityAsPartner("Entity")
```

```
HS.Entity.SecurityAsPartner("")
```

```
HS.Entity.SecurityAsPartner(Var1)
```

```
HS.Parent.SecurityAsPartner("Entity")
```



#### Note:

Use a blank string ("") to apply this function to the current entity member.

**Table 11-44 Syntax for SecurityAsPartner Function**

Parameter	Description
<i>Entity</i>	Name of a valid Entity dimension member.
<i>Var1</i>	A VisualBasic variable.

### Return Value

A string with the security class assigned to the entity or parent when it is used as an ICP.

### Example

In this example, if Class1 is the security class for France as it is used as an intercompany partner, then statements between the If...Then and If...End statements are executed:

```
If HS.Entity.SecurityAsPartner("France") = "Class1" Then
```

```
...
```

```
End If
```

```
If HS.Parent.SecurityAsPartner("France") = "Class1" Then
```

```
...
```

```
End If
```

## SecurityClass

Gets the security class for the specified member. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.<Object>.SecurityClass("Element")
```

```
HS.<Object>.SecurityClass("")
```

```
HS.<Object>.SecurityClass(Var1)
```

```
HS.Custom(Dimension).SecurityClass("Element")
```



#### Note:

Use a blank string ("") to apply this function to the current member.



**Table 11-45 Syntax for SecurityClass Function**

Parameter	Description
<Object>	One of these object keywords: <ul style="list-style-type: none"> <li>• Account</li> <li>• Scenario</li> <li>• Entity</li> <li>• Parent</li> <li>• Custom1...4</li> <li>• Custom (<i>Custom Dimension Label</i>)</li> </ul>
Element	Depending on the object selected, name of a valid member of one of these dimensions: <ul style="list-style-type: none"> <li>• Account</li> <li>• Scenario</li> <li>• Entity</li> <li>• Custom1...4</li> <li>• Custom</li> </ul>
Var1	A VisualBasic variable.

**Return Value**

The name of the security class assigned to the specified member.

**Example**

In this example, if Class1 is the security class assigned to the Cash account, then statements between the If...Then and End If statements are executed:

```
If HS.Account.SecurityClass("Cash") = "Class1" Then
```

```
...
```

```
End If
```

```
If HS.Custom(Dimension).SecurityClass(Member) Then
```

```
...
```

```
End If
```

## SetCellTextWithLabel

Updates the cell text information for a specified POV and cell text label. This function can be used in these types of rules:

Calculation

## Syntax

```
HS.SetCellTextWithLabel("POVExpression", "CellTextLabel", "CellTextString")
```

where `POVExpression` is a combination of members and `CellTextLabel` is either the default cell text label ([Default]) or one of the valid loaded cell text labels. `CellTextString` is the text string to be written. You must specify the Account, ICP, and Custom POV members. The Scenario, Year, Period, View, Entity, and Value default to the current members.

## Return Value

None

## Example

```
HS.SetCellTextWithLabel "A#Asset.I#[ICP  
None].C1#[None].C2#[None].C3#[None].C4#[None]", "Rating", "AAA"
```

# SetData

Sets an individual record. This function can be used in these types of rules:

- Calculation
- Translation

## Syntax

```
HS.SetData lView, lAccount, lICP, lCustom1, lCustom2, lCustom3, lCustom4,  
dData, bAddToExistingData
```

**Table 11-46** Syntax for SetData Function

Parameter	Description
<i>lView</i>	0 = Scenario View 1 = Periodic 2 = YTD
<i>lAccount</i>	ID number of the account to which you are setting data.
<i>lICP</i>	ID number of the ICP to which you are setting data.
<i>lCustom1...4</i>	ID number of the Custom dimension to which you are setting data.
<i>dData</i>	The data value to set.
<i>bAddToExistingData</i>	True = To accumulate the data False = To replace the data

## Return Value

None.

**Example**

```
HS.SetData 2, 002, , , , , 25000, TRUE
```

## SetDataWithPOV

Inserts data into the node or currency cube. This function can be used in these following types of rules:

- Calculation
- Translation

**Syntax**

```
HS.SetDataWithPOV POV, dData, bAddToExistingDataInCache
```

**Table 11-47 Syntax for SetData Function**

Parameter	Description
<i>POV</i>	Valid POV
<i>dData</i>	The data value to set.
<i>bAddToExistingData</i>	True = To accumulate the data False = To replace the data

**Return Value**

None.

**Example**

```
HS.SetDataWithPOV "W#YTD.A#Asset.I#[ICP  
None].C1#None.C2#None.C3#None.C4#None", 25000, TRUE
```

## SubmissionGroup

Gets the process management submission group for a dimension member.

**Syntax**

```
HS.<Dimension>.SubmissionGroup(Dimension member)
```

where *Dimension* is one of these dimensions: Account, Custom1...4, or Custom

```
HS.Account.SubmissionGroup(Account)
```

```
HS.Custom1.SubmissionGroup(Custom1)
```

```
HS.Custom2.SubmissionGroup(Custom2)
```

```
HS.Custom3.SubmissionGroup(Custom3)
```

```
HS.Custom4.SubmissionGroup(Custom4)
```

```
HS.Custom(Custom Dimension Label).SubmissionGroup((CustomDimensionLabel))
```

### Return Value

An integer representing the submission group for the dimension member. Valid values are 1–99.

### Example

```
HS.Account.SubmissionGroup("Sales")
```

```
HS.Custom("Product").SubmissionGroup("P3000-Phones")
```

## SupportsProcessManagement

Determines if a scenario supports process management. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.Scenario.SupportsProcessManagement("Scenario")
```

```
HS.Scenario.SupportsProcessManagement("")
```

**Note:**

Use a blank string ("") to apply this function to the current scenario.

```
HS.Scenario.SupportsProcessManagement(Var1)
```

**Table 11-48 Syntax for SupportsProcessManagement Function**

Parameter	Description
<i>Scenario</i>	A valid scenario.
<i>Var1</i>	VBScript variable representing a Scenario member.

**Return Value**

A Boolean that is True if the scenario has process management enabled; False otherwise.

**Example**

In this example, if process management is enabled for the actual scenario, statements between the If and End If statements are executed:

```
If HS.Scenario.SupportsProcessManagement("Actual") = "TRUE" then
```

```
...
```

```
End IF
```

## SupportsTran

Specifies the accounts in the application that support intercompany transactions. This function can be used only in Transactions rules.

**Syntax**

```
HS.SupportsTran "POVExpression"
```

where *POVExpression* is a combination of Account, Custom1-4, Scenario, Entity, and year members.

**Return Value**

None.

**Example**

```
HS.SupportsTran "S#ActMon.A#RecltIC.C1#Closing"
```

## SwitchSign

Determines if credits are switched to debits for the current Custom member or for a specified custom member. This function reverses the debit/credit sign using these rules:

- ASSET to LIABILITY
- LIABILITY to ASSET
- EXPENSE to REVENUE
- REVENUE to EXPENSE
- BALANCE to FLOW
- FLOW to BALANCE

This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.<Object>.SwitchSign("Member")
```

```
HS.<Object>.SwitchSign("")
```

```
HS.Custom(Dimension).SwitchSign(Member)
```



#### Note:

Use a blank string ("") to apply this function to the current member.

**Table 11-49 Syntax for SwitchSign Function**

Parameter	Description
<i>&lt;Object&gt;</i>	One of these object keywords: <ul style="list-style-type: none"> <li>• Custom1</li> <li>• Custom2</li> <li>• Custom3</li> <li>• Custom4</li> <li>• Custom</li> </ul>
<i>Member</i>	Name of a valid Custom dimension member.

### Return Value

A Boolean expression that is True if credits are switched to debits for the Custom member or False if credits and debits are not switched.

### Example

In this example, if the credits and debits are switched, then statements between the If...Then and End If statements are executed:

```
If HS.Custom1.SwitchSign("") = TRUE Then

    ...

End If

If HS.Custom("Product").SwitchSign("P3000-Phones") = TRUE Then

    ...

End If
```

## SwitchType

Determines if the account types are switched for the current Custom member or for a specified custom member. This function changes the account type for the Custom dimension member using these rules:

- ASSET to EXPENSE
- EXPENSE to ASSET
- LIABILITY to REVENUE
- REVENUE to LIABILITY
- BALANCE to FLOW
- FLOW to BALANCE

This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

## Syntax

```
HS.<Object>.SwitchType("Member")
```

```
HS.<Object>.SwitchType("")
```



### Note:

Use a blank string ("") to apply this function to the current member.

**Table 11-50 Syntax for SwitchType Function**

Parameter	Description
<i>&lt;Object&gt;</i>	One of these object keywords: <ul style="list-style-type: none"> <li>• Custom1</li> <li>• Custom2</li> <li>• Custom3</li> <li>• Custom4</li> <li>• Custom</li> </ul>
<i>Member</i>	Name of a valid Custom dimension member.

## Return Value

A Boolean expression that is True if account types are switched for the Custom member or False if account types are not switched.

## Example

In this example, if the account types are switched for the current Custom1 member, then statements between the If...Then and End If statements are executed:

```
If HS.Custom1.SwitchType("") = TRUE Then
```

```
...
```

```
End If
```

```
If HS.Custom("Product").SwitchType("P3000-Phones") = TRUE Then
```

```
...
```

```
End If
```



## Trans

Translates a currency using the year-to-date method. This function can be used in Translation rules.

### Syntax

```
HS.Trans("DestPOV","SourcePOV","Rate1","Rate2")
```

**Table 11-51 Syntax for Trans Function**

Parameter	Description
<i>DestPOV</i>	The destination point of view. The destination can be a combination of Account, Custom1...4, and ICP members. For each unspecified dimension, the system writes to all valid members of the dimension. For each specified dimension, the system writes into the specified member only.
<i>SourcePOV</i>	The source point of view. The source can be a combination of dimensions. If the Account, Custom1...4, and ICP dimensions are unspecified, the system uses the same member as the Destination member. If the Scenario, Year, Period, and Entity dimensions are not specified, the system uses the current members. If the Value dimension is not specified, the system uses the <EntityCurrTotal> member. If the source point of view is blank, the system uses the destination point of view as the source point of view.
<i>Rate1-2</i>	The exchange rate. The rate can be a constant, an exchange rate account, or a specific cell. Rate accounts are input for entity and for [None] entity. For information about the default translation process, see <a href="#">Default Translation</a> .

### Return Value

None.

### Example

This example uses the rate in the Rate1 account to translate the Sales account using the year to date method:

```
HS.Trans("A#Sales", "A#LastYearSales", "A#Rate1", "")
```

## TransPeriodic

Translates a currency using the periodic method. This function can be used in Translation rules.

## Syntax

```
HS.TransPeriodic("DestPOV", "SourcePOV", "Rate1", "Rate2")
```

**Table 11-52 Syntax for TransPeriodic Function**

Parameter	Description
<i>DestPOV</i>	The destination point of view. The destination can be a combination of Account, Custom1-4, and ICP members. For each not specified dimension, the system writes to all valid members of the dimension. For each specified dimension, the system writes into the specified member only.
<i>SourcePOV</i>	The source point of view. The source can be a combination of dimensions. If the Account, Custom1, Custom 2, Custom 3, Custom 4, and ICP dimensions are not specified, the system uses the same member as the Destination member. If the Scenario, Year, Period, and Entity dimensions are not specified, the system uses the current members. If the Value is not specified, the system uses the EntityCurrTotal member. If the source is blank, the system uses the destination as the source.
<i>Rate1-2</i>	The exchange rate. The exchange rate can be a constant, an exchange rate account, or a specific cell. Rate accounts are input for entity and for None entity. For information about the default translation process, see <a href="#">Default Translation</a> .

## Return Value

None

## Example

This example uses the exchange rate in the Rate1 account to translate the Sales account using the periodic method:

```
HS.TransPeriodic("A#Sales", "A#LastYearSales", "A#Rate1", "")
```

## UD1...3

Gets the text stored in the UserDefined1...3 attribute for the current member or for a specified member. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

## Syntax

```
HS.<Object>.UD1...3(strElement)
```

```
HS.<Object>.UD1...3("")
```

```
HS.Custom(Dimension).UD1(Member)
```



### Note:

Use a blank string ("" ) to apply this function to the current member.

**Table 11-53 Syntax for UD1...3 Functions**

Parameter	Description
<Object>	One of these object keywords: <ul style="list-style-type: none"><li>• Account</li><li>• Entity</li><li>• Parent</li><li>• Scenario</li><li>• Custom1...4</li><li>• Custom</li></ul>
Element	Depending on the object selected, name of a valid member of one of these dimensions: <ul style="list-style-type: none"><li>• Account</li><li>• Entity</li><li>• Parent</li><li>• Scenario</li><li>• Custom1...4</li><li>• Custom</li></ul>

## Return Value

A string that contains the user-defined text stored for the member.

### Example

In this example, if the user-defined text for the UD1 account is History, then statements between the If...Then and End If statements are executed.

```
If HS.Account.UD1(strAccount) = "History" Then

    ...

End If

If HS.Custom("Product").UD1("P3000-Phones") = "History" Then

    ...

End If
```

## ValidationAccount

Gets the validation account for an application. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.AppSettings.ValidationAccount
```

### Return Value

A string that specifies the name of the validation account for the application.

### Example

If the validation account for the application is MyAccount, then statements between the If...Then and the End If statements are executed.

```
If HS.AppSettings.ValidationAccount = "MyAccount" Then

    ...

End If
```

## ValidationAccountEx

Gets the validation account for a process management submission phase.

### Syntax

```
HS.AppSettings.ValidationAccountEx(n)
```

where *n* is an integer representing the process management submission phase. Valid values are 1 to 9.

### Return Value

A string that specifies the name of the validation account for the process management submission phase.

### Example

This example returns the validation account defined for Submission Phase 5:

```
HS.AppSettings.ValidationAccountEx(5)
```

## XBRLTags

Gets the XBRL tag assigned to the specified Account member. This function can be used in these types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```
HS.Account.XBRLTags("Account")
```

```
HS.Account.XBRLTags("")
```



#### Note:

Use a blank string ("") to apply this function to the current member.

**Table 11-54** Syntax for XBRLTags Functions

Parameter	Description
Account	A valid account.

**Return Value**

A string that specifies the XBRL tag for the specified account.

# 12

## Custom Functions

This section lists the internal HS custom functions available for Oracle Hyperion Financial Management for management reporting functions and planning functions. The functions include a description, type of function, syntax, example, and sample script.

**Table 12-1 Management Reporting Functions**

Custom Function	Description	Syntax	Function Type	Hyperion Enterprise Equivalent
Average	Calculates the financial average	Average (POV, Periods)	Function	AVE A12
Cumulative	Accumulates amounts from prior periods	Cumulative (POV, View, NumPeriod)	Function	CUM CTD YTD
Difference	Calculates the difference between current and opening	Difference (POV, View)	Function	DIF DFB
DSO	Calculates the days sales are outstanding	DSO (DSO, Debtor, Sales, DIP)	Procedure	Procedure
Opening	Carries opening balances forward	Opening (POV, View)	Function	OPE BASE BASEFLOW
Rate	Gets the relative exchange rate	Rate (ExchangeRate, Triangulation Currency)	Function	CrossRate

**Table 12-2 Planning Functions**

Custom Function	Description	Parameters	Function Type
Units_Rates	Units * rates (C=A*B)	Unit_Rates (Description, Units, Rates)	Procedure
Custom_Alloc	Allocates in the custom dimension	Custom_Alloc (Destination, Source, Factor, FactorN, FactorD, Elimination)	Procedure
Increase_Decrease	Increases or decreases the account by a percentage	Increase_Decrease (Destination, Source, Factor, Scale, Inverse)	Procedure
Pro_Rata_Ratio	Ratio between two accounts	Pro_Rata_Ratio (Destination, SourceN, SourceD)	Procedure
Spread	Spreads the total amount among all periods in the year	Spread (Destination, Source, Factor, FactorN, FactorD, Temp, Per)	Procedure

## Management Reporting Functions

This section lists the available management reporting custom functions.

## Average

Calculates the average value for an account across a number of periods.

### Return Value

Returns a string of characters representing the correct expression to be used as part of the HS.EXP function.

### Syntax

`Average (PointOfView, Periods)`

**Table 12-3 Syntax of Average Function**

Parameter	Valid Values
PointOfView	Valid combination of Account, Custom1....4, ICP members, for example, "A#CASH.C1#[None].I#[ICP Top]" For flow type accounts, the function averages only the periodic value.
Periods	Must be one of these values: YTD - Specify the year-to-date option to average the cumulative data from period one in the current year. Periodic - Specify the periodic option to average the current and immediately prior period in the current year only. For the first period, this value will be the same as the source. [Any whole positive number] - Specify a number of periods over which the average is to be calculated. For a rolling year average in a monthly category, specify 12.

### Detailed Description

This function calculates the average value of an account over a specified number of prior periods. If the source is a balance type account, the average is based on the entered data. If the source is a flow type account, the average is based on the periodic data only.

The Average value is derived differently based on the *Periods* parameter.

- If the *Periods* parameter is YTD, the average value is the sum of all periods in the current year up to the current divided by the current period number.
- If the *Periods* parameter is Periodic, the average value is the sum of the current and immediately prior periods divided by two. If the current period is the first period of the year, the average value is the same value as the source.
- If the *Periods* parameter is a number, the average value is the sum of the current and each preceding period for the specified number of periods, divided by the specified number.



**Example**

The SALES account will return these values for January, February, and March 2014 depending on the *Periods* parameter used in the Average custom function. The default view set for the scenario being processed is YTD.

**Table 12-4 Example of Average Function**

<b>Account</b>	<b>Oct2013</b>	<b>Nov2013</b>	<b>Dec2013</b>	<b>Jan2014</b>	<b>Feb2014</b>	<b>Mar2014</b>
A#Sales	9,000	10,500	11,700	800	1,900	3,200
Average ("A#Sales", "YTD")	N/A	N/A	N/A	800	950	1,067
Average ("A#Sales", "Periodic")	N/A	N/A	N/A	800	950	1,200
Average ("A#Sales", "3")	N/A	N/A	N/A	1,167	1,033	1,067

**Sample Script**

```
' sample statement written in the calling routine
```

```
Sub Calculate()
```

```
Hs.Exp "A#AVG_SALES = "& Average("A#Sales", "12")
```

```
End Sub
```

```
' programming of the AVERAGE function
```

```
FUNCTION Average(strPOV,strPERIOD)
```

```
  DIM nPERIOD
```

```
  DIM strCUM
```

```
  DIM i
```

```
  strPOV = UCASE(strPOV)
```

```
  strPERIOD = UCASE(strPERIOD)
```

```
  IF strPERIOD = "PERIODIC" THEN
```

```
    IF HS.PERIOD.ISFIRST = TRUE THEN
```

```
      nPERIOD = 1
```

```
    ELSE
```

```
      nPERIOD = 2
```

```
    END IF
```

```
  ELSEIF strPERIOD = "YTD" THEN
```

```
    nPERIOD = HS.PERIOD.NUMBER()
```

```
  ELSEIF CINT(strPERIOD) > 0 THEN
```

```
    nPERIOD = CINT(strPERIOD)
```

```
  ELSE
```

## Cumulative

Calculates the total of the preceding period's values for a specified account.

### Return Value

Returns a string of characters representing the correct expression to be used as part of the HS.EXP function.

### Syntax

`Cumulative (PointOfView, View, NumPeriod)`

**Table 12-5 Syntax of Cumulative Function**

Parameter	Valid Values
PointOfView	Valid combination of Account, Custom1....4, ICP members, for example, "A#CASH.C1#[None].I#[ICP Top]"
View	Must be one of these values: "" " " (double quote) - Based on the default view defined for the scenario being processed (either YTD or Periodic). YTD - User specifies the Year-to-date option, which overrides the default view set for the scenario. Periodic - Specify the periodic option, which overrides the default view set for the scenario.
NumPeriod	A whole number representing the number of periods in the current scenario to accumulate, starting with the current period.  If NumPeriod is 0 or negative, the function aggregates from the beginning of the current year.

### Detailed Description

This function calculates the sum of either the periods specified or the sum year to date for the specified account. By default, the view of the accumulated data is the scenario default; however, you can override this for flow type accounts.

- If the *View* parameter is YTD, the function accumulates the year-to-date values.
- If the *View* parameter is Periodic, the function accumulates the periodic values.
- If the *View* parameter is blank ( " " ), the function accumulates the data using the scenario default view.

### Example

The CASH account will return the following values for January, February, and March 2014 depending on the *Number* parameter used in the Cumulative function.

The a SALES account will return the following values for January, February, and March 2014 depending on both the *View* and *Number* parameters used in the Cumulative function. The default view set for the scenario being processed is YTD.

**Table 12-6 Example of Cumulative Function**

Account	Oct2013	Nov2013	Dec2013	Jan2014	Feb2014	Mar2014
A#Cash	1,000	1,500	1,200	800	1,100	1,300
Cumulative ("A#Cash", "0")	N/A	N/A	N/A	800	1,900	3,200
Cumulative("A# Cash", "3")	N/A	N/A	N/A	3,500	3,100	3,200
A#Sales	9,000	10,500	11,700	800	1,900	3,200
Cumulative("A# Sales", "0")	N/A	N/A	N/A	800	2,700	5,900
Cumulative("A# Sales", "Periodic", "0")	N/A	N/A	N/A	800	1,900	3,200
Cumulative("A# Sales", "Periodic", "3")	N/A	N/A	N/A	3,500	3,100	3,200

**Sample Script**

```
' sample statement written in the calling routine
```

```
Sub Calculate()
```

```
HS.EXP "A#TOT_Cash =" &Cumulative("A#Cash","" ,0)
```

```
End Sub
```

```
' programming of the Cumulative function
```

```
Function Cumulative(StrPov, StrVIEW, nPERIOD)
```

```
  DIM strCUM
```

```
  DIM i
```

```
  IF nPERIOD <= 0 THEN
```

```
    nPERIOD = HS.PERIOD.NUMBER() - 1
```

```
  ELSE
```

```
    nPERIOD = nPERIOD - 1
```

```
  END IF
```

```
  IF strVIEW = "" THEN
```

```
    strVIEW = HS.SCENARIO.DEFAULTVIEW("")
```

```
  END IF
```

```
  strPOV = UCASE(strPOV)
```

```
  strVIEW = UCASE(strVIEW)
```

```
  IF strVIEW = "PERIODIC" THEN
```

```
    strVIEW = ".W#PERIODIC"
```

## Difference

Calculates the difference between the current period value and the opening value.

### Return Value

Returns a string of characters representing the correct expression to be used as part of the HS.EXP function.

### Syntax

`Difference (PointOfView, View)`

**Table 12-7 Syntax of Difference Function**

Parameter	Valid Values
PointOfView	Valid combination of Account, Custom1....4, ICP members, for example, "A#CASH.C1#[None].I#[ICP Top]"
View	Must be one of these values: "" (double quote) - Based on the default view defined for the scenario being processed (either YTD or Periodic). YTD - Specify the Year-to-date option, which overrides the default view set for the scenario. Periodic - Specify the periodic option, which overrides the default view set for the scenario.

### Detailed Description

This function calculates the difference between the value of the current period and the opening value. (Current - Opening)

The opening value is derived differently based on the *View* parameter passed to the function.

- If the *View* parameter is YTD, the opening value is retrieved from the last period of the prior year.
- If the *View* parameter is Periodic, the opening value is retrieved from the prior period of the current year. If the current period is the first period of the year, the opening value is retrieved from the last period of the prior year.
- If the *View* parameter is blank ( " " ), the opening value is based upon the default data view of the scenario.

### Example

The CASH account will return the following values for January, February, and March 2014 depending on the *View* parameter used in the Difference function. The default view set for the scenario being processed is YTD. The Difference function subtracts the opening value from the current period value.

**Table 12-8 Example of Difference Function**

<b>Account</b>	<b>Dec2013</b>	<b>Jan2014</b>	<b>Feb2014</b>	<b>Mar2014</b>
A#Cash	900	1,200	1,100	1,500
Difference("A#Cash", ""	N/A	300	200	600
Difference("A#Cash", "YTD")	N/A	300	200	600
Difference("A#Cash", "Periodic")	N/A	300	-100	400

**Sample Script**

```
' sample statement written in the calling routine

Sub Calculate()

Hs.Exp = "A#DiffCash" & Difference("A#Cash", "YTD")

End Sub

' programming of the DIFFERENCE function

FUNCTION DIFFERENCE(strPOV,strVIEW)

IF strVIEW = "" THEN

strVIEW = HS.SCENARIO.DEFAULTVIEW ("" )

END IF

strPOV = UCASE(strPOV)
strVIEW = UCASE(strVIEW)

IF strVIEW = "PERIODIC" THEN

DIFFERENCE = ""&" "&strPOV"-& strPOV & ".P#PRIOR" &""""

ELSEIF strVIEW = "YTD" THEN

DIFFERENCE = "" (&strPOV &""-" & strPOV & ".Y#PRIOR.P#LAST" &"" ) ""

ELSE

EXIT FUNCTION

END IF

END FUNCTION
```



## DSO - Days Sales Outstanding

Calculates the number of days sales in the current period debtors using the exhaustion method.

### Return Value

This routine calculates a single value representing the amount of days sales contained within the current period trade debtors figure. The DSO sub-routine included makes these assumptions:

- Both Debtors and Sales are positive figures.
- The parameters supplied are fully defined points of view (for example, Account/C1/C2/C3/C4/ICP) because the routine uses the HS.GETCELL function.
- The routine calculates the days going back as far as possible in time. However, it will stop if the periodic sales value for any period is a negative or zero value.

### Syntax

```
CALL DSO (strDSO,strDEBTOR,strSALES,strDIP)
```

**Table 12-9 Syntax of DSO Function**

Parameter	Valid Values
strDSO	Fully defined account with custom and intercompany dimensions. This account is the destination for the calculation.
strDEBTOR	Fully defined account with custom and intercompany dimensions. This account is the source for the current period trade debtors.
strSALES	Fully defined account with custom and intercompany dimensions. This account is the source for the sales. Specifically exclude references to frequency.
strDIP	Fully defined account with custom and intercompany dimensions. This account is the source for the number of days in the period. This is assumed to be in the [None] entity.

### Detailed Description

The routine takes the values in the Debtors account (parameter 2) and Sales account (parameter 3) for the current period and compares them. If either are zero or negative, the calculation stops. For each successive period where the debtors value exceeds that of the cumulative sales (working backwards from the current period), the routine adds the number of days for that period as specified in the days in the Period account (parameter 4) to a running total.

When all the Debtors values have been exhausted in this way, the final period's days are calculated as a proportion of the unexpired debtors against the periodic sales value.

Finally, the routine posts the running total to the destination account (parameter 1).

**Example**

The example calculates the total days outstanding for the months shown.

**Table 12-10 Example of DSO Function**

Month	Debtors	Period Sales	Days in Month	Formula for DSO	Total DSO
September	12,000	2,500	30	100%	30
August	N/A	1,750	31	100%	31
July	N/A	2,250	31	100%	31
June	N/A	2,500	30	100%	30
May	N/A	2,000	31	100%	31
April	N/A	2,250	30	2000/2250	26.7
Total	N/A	N/A	N/A	N/A	179.7

**Sample Script**

```
' Use within the calculation section:
' 1. Standard use

CALL DSO("A#DSO", "A#TradeDebtors.C1#AllAges.C2#[None].I#[ICP
Top", "A#TotalSales.C1#[None].C2#AllProducts.I#[ICP Top]", "A#DIP")

' 2. Use with a common custom dimension

set vPRODUCT = ARRAY("C2#PRODUCT1", "C2#PRODUCT2", .... , "C2#PRODUCTn")

FOR EACH iITEM IN vPRODUCT

CALL DSO(" " A#DSO. " "&iITEM, " " A#TradeDebtors.C1#AllAges.I#[ICP
Top]. " "&iITEM, " " A#TotalSales.C1#[None].I#[ICP Top]. " " &iITEM, " " A#DIP")

NEXT

' Actual script of Sub-routine

SUB DSO(strDSO, strDEBTOR, strSALES, strDIP)

DIM vTEST
DIM vDSO
DIM vCOUNT
```

```
DIM vXS_1
DIM vXS

HS.CLEAR(strDSO)

vTEST = HS.GETCELL(strDEBTOR) * HS.GETCELL(strSALES&"".W#Periodic") *
HS.GETCELL(strDIP&"".E#[None]")

' checks if any of the parameters are zero (uses principle of X * 0 = 0)

IF vTEST = 0 THEN

EXIT SUB

ELSE

vDSO = 0
vCOUNT = 0
vXS_1 = HS.GETCELL(strDEBTOR)
vXS = vXS_1 - HS.GETCELL(strSALES&"".W#Periodic")

' ensures that periodic sales are not negative or zero

WHILE vXS > 0 AND vXS_1 > vXS

vDSO = vDSO + HS.GETCELL(strDIP&"".E#[None].P#CUR-"" &vCOUNT)
vCOUNT = vCOUNT + 1
```

```
vXS_1 = vXS
vXS = vXS - HS.GETCELL(strSALES&"".W#Periodic.P#CUR-"" &vCOUNT)

WEND

IF vXS = vXS_1 THEN

vCOUNT = vCOUNT - 1

END IF

vDSO = vDSO + (vXS_1 / HS.GETCELL(strSALES&"".W#Periodic.P#CUR-""
&vCOUNT)*HS.GETCELL(strDIP&"".E#[None].P#CUR-"" &vCOUNT))

IF vDSO < 0 THEN

vDSO = 0

END IF

END IF

HS.EXP strDSO &"="& vDSO

END SUB
```

## Opening

Retrieves the opening value for a specified, fully defined account (Account/C1/C2/C3/C4/ICP).

### Return Value

This function returns a string of characters representing the correct expression to be used as part of the HS.EXP function.

### Syntax

Opening (*PointOfView*, *View*)

**Table 12-11 Syntax of Opening Function**

Parameter	Valid Values
PointOfView	Valid combination of Account, Custom1....4, ICP members, for example, "A#CLOSE.C1#[None].I#[ICP Top]"
View	<p>Must be one of these values:</p> <p>"" (double quote) - Based on the default view defined for the scenario being processed (either YTD or Periodic).</p> <p>YTD - Specify the Year-to-date option, which overrides the default view set for the scenario.</p> <p>Periodic- Specify the Periodic option, which overrides the default view set for the scenario.</p>

**Detailed Description**

This function calculates the opening value of a specified account. The opening value is derived differently based on the *View* parameter.

- If the *View* parameter is YTD, the opening value is retrieved from the last period of the prior year.
- If the *View* parameter is Periodic, the opening value is retrieved from the prior period of the current year. If the current period is the first period of the year, the opening value is retrieved from the last period of the prior year.
- If the *View* parameter is blank ( " " ), the opening value is based on the default data view of the scenario.

**Example**

The FA\_COST account will return the following values for January, February, and March 2014 depending on the *View* parameters used in the Opening function. The default view set for the scenario being processed is YTD.

**Table 12-12 Example of Opening Function**

Account	Dec2013	Jan2014	Feb2014	Mar2014
A#FA_COST	900	1,200	1,100	1,500
Opening("A#FA_COST ", "")	N/A	900	900	900
Opening("A#FA_COST ", "YTD")	N/A	900	900	900
Opening("A#FA_COST ", "Periodic")	N/A	900	1,200	1,100

### Sample Script

```
' sample statement written in the calling routine
```

```
Sub Calculate()
```

```
  Hs.Exp "A#Open_FA_Cost = " & Opening("A#FA_Cost", "YTD")
```

```
End Sub
```

```
' programming of the OPENING function
```

```
FUNCTION OPENING(strPOV,strVIEW)
```

```
  IF strVIEW = "" THEN
```

```
    strVIEW = HS.SCENARIO.DEFAULTVIEW ( "")
```

```
  END IF
```

```
  strPOV = UCASE(strPOV)  
  strVIEW = UCASE(strVIEW)
```

```
  IF strVIEW = "PERIODIC" THEN
```

```
    OPENING = strPOV & ".P#PRIOR"
```

```
  ELSEIF strVIEW = "YTD" THEN
```

```
    OPENING = strPOV & ".Y#PRIOR.P#LAST"
```

```
  ELSE
```

```
    EXIT FUNCTION
```

```
  END IF
```

```
END FUNCTION
```

# Rate

Calculates the relative exchange rate between a parent and child and returns the value as a multiplier.

## Return Value

This function returns a value to be used as part of an HS.EXP function, usually in the translation section.

## Syntax

Rate (*ExchangeRate*, *TriangulationCurrency*)

**Table 12-13 Syntax of Rate Function**

Parameter	Valid Values
ExchangeRate	A main account of the "CurrencyRate" type specified as an account string, without reference to custom or intercompany dimensions, for example, "A#EOP_RATE"
TriangulationCurrency	Either a valid currency label as a string or double quotes ( " "). When specifying a currency, it is not necessary to reference any custom dimension.

## Detailed Description

- This function calculates the relative exchange rate between a parent and child, returning a value as a multiplier. The value is calculated based on the *TriangulationCurrency* parameter.
- If the *TriangulationCurrency* parameter is a valid currency label, the cross rate is based on this currency.
- If the *TriangulationCurrency* parameter is blank ( " " ), the function first searches for a valid direct rate, and if none is found, uses Triangulation against the application currency.
- If no rate values are found, the function returns 1.

These tables show the methods of searching for the data and the order in which the search is made. The order is represented by a number in parentheses, for example (1). In each case, the search is made first in the child entity and, if no data is found, then from the "[None]" entity.

In the following table, either the currency of the child or of the parent is the same as the Triangulation currency, or if Triangulation is blank, the application currency.

**Table 12-14 Rate Example — Triangulation Currency Same**

Custom 2 dimension rates		Custom 1 dimension rates	
		Child	Parent
Custom 2 dimension rates	Child		(2)
Parent	(1)		

In the following table, Triangulation has been specified and is not the same as either the child or parent currencies.

**Table 12-15 Rate Example — Triangulation Currency Different**

Custom 2 dimension rates		Custom 1 dimension rates		
		Child	Parent	Triangulation
Custom 2 dimension rates	Child			(2)
	Parent			
	Triangulation	(1)		

In the following table, Triangulation has not been specified and the application currency is different from both the child and parent currencies.

**Table 12-16 Rate Example — Triangulation Not Specified**

Custom 2 dimension rates		Custom 1 dimension rates		
		Child	Parent	Application
Custom 2 dimension rates	Child		(2)	(4)
	Parent	(1)		
	Application	(3)		

### Example

The application currency is Euros, and you need to translate a French child entity to a US parent entity using these rates entered in the [None] entity against the C2#EURO:

**Table 12-17 Example of Rate Function**

Rate	Opening Rate	Closing Rate
C1#FFR	0.16000	0.16500
C1#USD	1.15862	1.15785

The following function multiplies the opening balance account by the difference between the relative ending and opening rates. This is useful when calculating movement analyses if the translation is not consistent between the local and application currencies.

```
HS.EXP "A#FXO = A#OPEN * (" & RATE("A#EOP_RATE", " ") & "-" &
RATE("A#OPE_RATE", " ") &")"
```

For the previous example, if the value in the OPEN account for the child is FFR 10,000,000, the value in the US parent FXO account will be USD 44,102 [10,000,000 \* (0.165 /1.15785 - 0.16 /1.15862)].



## Sample Script

```
' sample statement written in the calling routine

SUB TRANSLATE()

HS.TRANS "A#FXO", "A#FXO", "A#EOP_RATE", ""
HS.EXP "A#FXO = A#OPEN * (" & RATE("A#EOP_RATE", " ") & "-" &
RATE("A#OPE_RATE", " ") &")"

END SUB

' programming of the RATE function

FUNCTION RATE(sRATE, sTRI)

DIM sCCUR, sPCUR, sACUR, bRET, retValue, s3rdCUR
DIM i

sRATE = UCASE(sRATE)
sTRI = UCASE(sTRI)
sCCUR = UCASE(HS.ENTITY.DEFCURRENCY(""))
sPCUR = UCASE(HS.VALUE.CURRENCY)
```

```

sACUR = UCASE(HS.APPSETTINGS.CURRENCY)
retValue = 0

' check whether there is a triangulation specified, or if triangulation or
application currencies are the same as either parent or child and set up the
select case

IF sTRI = sCCUR OR sTRI = sPCUR OR (sTRI = " " AND (sACUR = sCCUR OR sACUR =
sPCUR)) THEN

i = 1

ELSEIF sTRI <> " " THEN

i = 2

ELSE

i = 3

END IF

SELECT CASE i

CASE 1

' bRET is a boolean that returns true if data is found. First search the
child...
' ...then search the [None] entity

bRET = GETVALUECP(".V#<Entity Currency>",retValue,sRATE,sCCUR,sPCUR)

IF NOT bRET THEN

bRET = GETVALUECP(".E#[None]",retValue,sRATE,sCCUR,sPCUR)

END IF

CASE 2

' use a dynamic parameter name for ease of writing the triangulation checks

```

```

sCCUR,sPCUR,s3rdCUR)

END IF

CASE 3

' this case is used when the 2nd parameter is blank and is the most complex.

' first check direct rates in the child...

' ... then check triangulation against application currency in the child

' then check direct rates in [None].

'... finally check triangulation in [None]

s3rdCUR = sACUR

bRET = GETVALUECP(".V#<Entity Currency>",retValue,sRATE,sCCUR,sPCUR)

IF NOT bRET THEN

bRET = GETVALUE3(".V#<Entity Currency>",retValue,sRATE,sCCUR,sPCUR,s3rdCUR)

IF NOT bRET THEN

bRET = GETVALUECP(".E#[None]",retValue,sRATE,sCCUR,sPCUR)

IF NOT bRET THEN

bRET = GETVALUE3(".E#[None]",retValue,
sRATE,sCCUR,sPCUR,s3rdCUR)

END IF

END IF

END IF

END SELECT

```

```

parent

GETVALUECP = FALSE

' check if data exists for direct rate child to parent. If it does return it.
' if no direct child to parent rate check for indirect parent to child rate...
' return the inverse of the indirect rate.

IF HS.GETCELL(sRATE & ".C1#" & sCCUR & ".C2#" & sPCUR & sENTITY) <> 0 THEN

sVALUE = CDBL(HS.GETCELL(sRATE & ".C1#" & sCCUR & ".C2#" & sPCUR & sENTITY))

GETVALUECP = TRUE

ELSEIF HS.GETCELL(sRATE & ".C1#" & sPCUR & ".C2#" & sCCUR & sENTITY) <> 0 THEN

sVALUE = CDBL(1 / HS.GETCELL(sRATE & ".C1#" & sPCUR & ".C2#" & sCCUR &
sENTITY))

GETVALUECP = TRUE

END IF

END FUNCTION

FUNCTION GETVALUE3(sENTITY, sVALUE, sRATE, sCCUR, sPCUR, s3rdCUR)

' this sub-function is used when triangulating
' check if data exists for direct rate child to triangulation...
' ... if it does return the direct relative rate child to parent...
' if no direct child to triangulation rate check for indirect triangulation

```

```
to child rate..
' ... return the inverse of the indirect relative rates.

GETVALUE3 = FALSE

IF HS.GETCELL(sRATE & ".C1#" & sCCUR & ".C2#" & s3rdCUR & sENTITY) <> 0 THEN

sVALUE = CDBL(HS.GETCELL(sRATE & ".C1#" & sCCUR & ".C2#" & s3rdCUR &
sENTITY) / HS.GETCELL(sRATE & ".C1#" & sPCUR & ".C2#" & s3rdCUR & sENTITY))

GETVALUE3 = TRUE

ELSEIF HS.GETCELL(sRATE & ".C1#" & s3rdCUR & ".C2#" & sCCUR & sENTITY) <> 0
THEN

sVALUE = CDBL(HS.GETCELL(sRATE & ".C1#" & s3rdCUR & ".C2#" & sPCUR &
sENTITY) / HS.GETCELL(sRATE & ".C1#" & s3rdCUR & ".C2#" & sCCUR & sENTITY))

GETVALUE3 = TRUE

END IF

END FUNCTION
```

## Business Rules Functions

This section lists business rules custom functions.

### Custom\_Alloc

This function allocates a Source point of view (POV) to a Destination POV using a Factor POV as the basis of Allocation, with the option to reverse post the total allocated amount to an Elimination POV. This function is designed for custom dimension allocations.

#### Return Value

No return value.

#### Syntax

```
Custom_Alloc(Destination, Source, Factor, FactorN, FactorD,  
Elimination)
```

**Table 12-18 Syntax of Custom\_Alloc Function**

Parameter	Valid Values
Destination	A valid destination POV that is a valid combination of Account, ICP and Custom 1-4 members.
Source	A valid source POV that is a valid combination of dimension members. <i>Source</i> is the amount that is to be allocated.
Factor	A valid source POV. <i>Factor</i> is the Account used to store the allocation factor.
FactorN	A valid source POV. <i>FactorN</i> is the numerator factor used as the basis for allocation.
FactorD	A valid source POV. <i>FactorD</i> is the denominator factor used as the basis for allocation.
Elimination	A valid source POV. <i>Elimination</i> can be an empty string (""), in which case this parameter is ignored. If the <i>Elimination</i> parameter is set, the amount posted to the <i>Destination POV</i> is multiplied by -1 and posted to the Elimination POV.

### Detailed Description

This function allocates a Source POV to a Destination POV using a Factor POV as the basis of allocation, with the option to reverse post the total allocated amount to an Elimination POV. This function is designed for custom dimension allocations.

The *Factor* parameter stores the result of *FactorN* divided by *FactorD*. This is required to enable the factor to refer to entities other than the current entity.

If the entity in the Source POV is a parent, that parent must be consolidated before executing the calculation at the child level. If the parent currency is different from the child currency, then a translation of all relevant currencies must also be run before executing the calculation at the child level.

It is recommended that variables are set in the calling routine and passed to the Custom\_Alloc function, which define the Destination, Source, Factor, FactorN, FactorD and Elimination POVs. It is also recommended that the variable names in the calling routine be set to be the same as the Custom\_Alloc function.

The *Elimination* parameter can be an empty string (""), in which case this parameter is ignored. If the *Elimination* parameter is set, the amount posted to the Destination POV will be multiplied by -1 and posted to the Elimination POV.

### Example

The Telephone account is allocated to Products based on a ratio of Products Sales to Total Sales. The inverse of the allocated amount is posted to Allocations account.

**Table 12-19 Example of Custom\_Alloc Function**

Account	Jan2014	Feb2014	Mar2014
A#Telephone.C1#[None]	100	300	400
A#Sales.C1#Product1	1000	1000	1000

**Table 12-19 (Cont.) Example of Custom\_Alloc Function**

Account	Jan2014	Feb2014	Mar2014
A#Sales.C1#Product2	1000	2000	3000
A#Sales.C1#TotalProducts	2000	3000	4000
Custom_Alloc(A#Telephone ", "A#Telephone.C1#[None ]", "A#Factor", A#Sales", "A#Sales.C1#TotalProduc ts", "A#ProductAllocations.C 1#[None])	N/A	N/A	N/A
A#Factor.C1#Product1	0.50	0.33	0.25
A#Factor.C1#Product2	0.50	0.66	0.75
A#Telephone.C1#Product1	50	100	100
A#Telephone.C1#Product2	50	200	300
A#ProductAllocations.C1#[No ne]	-100	-300	-400

The result returned from the CUSTOM\_ALLOC function is as follows:

```
HS.EXP "A#Factor = A#Sales / A#Sales.C1#TotalProducts"
```

```
HS.EXP "A#Telephone = A#Telephone.C1#[None] * A#Factor"
```

```
HS.EXP "A#Allocations.C1#[None] = (A#Telephone.C1#[None] * -1)"
```

### Sample Script

This script contains the following information:

- A sample statement written in the calling routine.
- Variables set in the calling routine and passed to the Custom\_Alloc function.
- Variable names in the calling routine set to be the same as the Custom\_Alloc function.

```
Sub Calculate()
```

```
Dim Destination
Dim Source
Dim Elimination
Dim Factor
Dim FactorN
Dim FactorD
```

```
Dim C1list
Dim Clitem

C1list = HS.Custom1.List("Alloc")

For Each Clitem in C1list

    Source = "A#Telephone.C1#[None]"
    Destination = "A#Telephone.C1#" & Clitem
    Factor = "A#Factor.C1#" & Clitem
    FactorN = "A#Sales.C1#" & Clitem
    FactorD = "A#Sales.C1#TotalProducts"
    Elimination = "A#ProductAllocations.C1#" & Clitem

    Call Custom_Alloc(Destination,Source,Factor,FactorN,
    FactorD,Elimination)

Next

End Sub

' Beginning of the Custom_Alloc function

Sub Custom_Alloc(Destination,Source,FactorN,FactorD,
Elimination)

HS.Clear Factor

HS.Exp Factor & " = " & FactorN & "/" & FactorD

HS.EXP Destination & " = " & Source & " * " & Factor

If Elimination <> "" Then

HS.EXP Elimination & " = " & Source & " * -1 * " & Factor

End If

End Sub
```



## Increase\_Decrease

This function increases or decreases a Destination POV by a percentage Factor. The percentage factor may be taken from either a Source POV, a VBScript constant or a VBScript variable.

### Return Value

No return value.

### Syntax

`Increase_Decrease (Destination, Source, Factor, Scale, Inverse)`

**Table 12-20 Syntax of Increase\_Decrease Function**

Parameter	Valid Values
Destination	A valid destination POV that is a valid combination of Account, ICP and Custom 1-4 members.
Source	A valid source POV that is a valid combination of dimension members. <i>Source</i> is the amount that is to be allocated.
Factor	A valid source POV, constant, or variable.
Scale	Integer value 1 or 100. Factor is divided by scale.
Inverse	True or False. True reverses the sign of the Factor. This can be used to generate a decrease where the Factor is stored as a positive number (or vice-versa). False uses the stored sign of the Factor to determine an increase or decrease.

### Detailed Description

This function increases or decreases a Destination POV by a percentage factor. The percentage factor may be taken from a Source POV, a VBScript constant or a VBScript variable.

In general, the Source POV is the same as the Destination POV, however, it can be different.

The *Scale* parameter is used to scale down the factor, if required. This applies when the factor is taken from a Source POV and the factor is stored in a non-scaled form (for example, 50% is stored as 50 and not 0.50).

The *Inverse* parameter is used to reverse the sign of the factor. This applies when the factor is taken from a Source POV and the factor is stored as an absolute number. If the *Inverse* parameter is set to True, the factor is multiplied by -1. If the *Inverse* parameter is set to False, the factor is not multiplied -1.

### Example

In this example, the Telephone account is increased by 10%.

**Table 12-21 Example of Increase\_Decrease Function**

Account	Jan2014	Feb2014	Mar2014
A#Telephone	100	300	400
A#Factor/C1[None]	10	10	10
	N/A	N/A	N/A
Increase_Decrease("A# Telephone", "A#Telephone", "A#Factor.C1#[None]", 100,False)			
A#Telephone	110	330	440

The result returned from the INCREASE\_DECREASE function is as follows:

```
HS.EXP "A#Telephone = A#Telephone * (1+ (A#Factor.C1#[None]/100))"
```

### Sample Script

- A sample statement written in the calling routine.
- Variables set in the calling routine and passed to the Increase\_Decrease function.
- Variable names in the calling routine set to be the same as the Increase\_Decrease function.

```
Sub Calculate()
```

```
Dim Destination
Dim Source
Dim Factor
Dim Scale
Dim Inverse
```

```
Destination = "A#Telephone"
Source = "A#Telephone"
Factor = "A#Factor.C1#[None]"
```

```
Scale = "100"
Inverse = False

Call Increase_Decrease(Destination,Source,Factor,Scale,
Inverse)

End Sub

' Beginning of the Increase_Decrease function

Sub Increase_Decrease(Destination,Source,Factor,Scale,Inverse)

If Inverse = False Then

HS.EXP Destination & " = " & Source & " *

(1 + (" & Factor & " / " & Scale & "))"

Else

HS.EXP Destination & " = " & Source & " *

(1 + ((" & Factor & " * -1) / " & Scale & ))"

End If

End Sub
```

## Pro\_Rata\_Ratio

This function calculates the ratio between two source POVs ( $C = A / B$ ).

### Return Value

No return value.

### Syntax

```
Pro_Rata_Ratio(Destination,SourceN,SourceD)
```

**Table 12-22 Syntax of Pro\_Rata\_Ratio Function**

Parameter	Valid Values
Destination	A valid destination POV that is a valid combination of Account, ICP and Custom 1-4 members.
SourceN	A valid source POV that is a valid combination of dimension members. <i>SourceN</i> is the numerator of the ratio calculation.
SourceD	A valid source POV. <i>SourceD</i> is the denominator of the ratio calculation.

**Detailed Description**

This function calculates the ratio between two source POVs ( $C = A / B$ ).

As a best practice, it is recommended that variables are set in the calling routine and passed to the Pro\_Rata\_Ratio function, which define the Destination, SourceN and SourceD POVs. It is also recommended that the variable names in the calling routine be set to be the same as the Pro\_Rata\_Ratio function.

The system does not naturally calculate weighted average ratios for parent members. Parent member values display as an aggregation of child values. This results in a mathematically incorrect value for parent members. As such, it is recommended that aggregation be turned off for Ratio accounts.

**Example**

The MarginPct account will return the value of GrossMargin/TotalRevenues.

**Table 12-23 Example of Pro\_Rata\_Ratio Function**

Account	Jan2014	Feb2014	Mar2014
A#GrossMargin	1000	100	750
A#TotalRevenues	2000	400	1000
	0.50	0.25	0.75

```

Pro_Rata_Ratio("A
#GrossMargin", "#T
otalRevenues")

```

The result returned from the PRO\_RATA\_RATIO function is as follows:

```
HS.EXP "A#MarginPct = A#GrossMargin / A# TotalRevenues"
```

**Sample Script**

The script contains the following information:

- A sample statement written in the calling routine.
- Variables set in the calling routine and passed to the Pro\_Rata\_Ratio function.

- Variable names in the calling routine set to be the same as the Pro\_Rata\_Ratio function.

```

Sub Calculate()

Dim Destination 'Destination POV
Dim SourceN     'Source Numerator POV
Dim SourceD     'Source Denominator POV

Destination = "A#MarginPct"
SourceN = "A#GrossMargin"
SourceD = "A#TotalRevenues    "

Call Pro_Rata_Ratio(Destination,SourceN,SourceD)

End Sub

' Beginning of the Pro_Rata_Ratio function

Sub Pro_Rata_Ratio(Destination,SourceN,SourceD)

HS.EXP Destination & " = " & SourceN & " / " & SourceD

End Sub

```

## Spread

This function allocates a single time period value (for example, P#[Year]) of a Source Account to all periods of a Destination Account based on a profile defined in a Profile Account (for example, Revenue profile, 4-4-5, etc.).

### Return Value

No return value.

### Syntax

```
Spread(Destination,Source,Factor,FactorN,FactorD,Temp,Per)
```

**Table 12-24 Syntax of Spread Function**

Parameter	Valid Values
Destination	A valid destination POV that is a valid combination of Account, ICP and Custom 1-4 members.

**Table 12-24 (Cont.) Syntax of Spread Function**

Parameter	Valid Values
Source	A valid source POV that is a valid combination of dimension members. The Source POV must include a single time period, for example, P#[Year]. The single time period amount is the amount to be spread.
Factor	A valid source POV. <i>Factor</i> is the account used to store the allocation factor.
FactorN	A valid source POV. <i>FactorN</i> is the numerator factor used as the basis for spread allocation.
FactorD	A valid source POV. <i>FactorD</i> is the denominator factor used as the basis for spread allocation.
Temp	A valid destination Account. <i>Temp</i> is the account that temporarily stores the Source value.
Per	A period string that defines the name of the first period in the timeframe, for example, "January". The <i>Temp</i> value is stored in the first period and the parameter is required to refer to this in the calculation.

**Detailed Description**

This function allocates a single time period value (for example, P#[Year]) of a Source POV to all periods of a Destination POV based on a profile defined in a Profile POV (for example, Revenue profile, 4-4-5, and so on).

Time-based allocations are particularly suited to budgeting applications where amounts are first entered for the total year, and then later allocated across time periods based on a suitable profile.

The Source POV must contain a single time period. The time period will generally be P#[Year], but could be any single period, such as P#January.

The value in the Source POV is stored by the calculation in a temporary account. This is required because the source and destination accounts are typically the same account. Where this is the case, the value in P#[Year] changes as the calculation proceeds from 1 period to the next. Therefore, you must store the value first to be able to refer to it for all time periods.

It is recommended that variables are set in the calling routine and passed to the Spread function, which define the Destination, Source, Profile, Temp, and Period1 parameters. It is also recommended that the variable names in the calling routine be set to be the same as the Spread function.

**Example**

The Year value in the Telephone account are allocated across Time Periods using a 4-4-5 quarterly ratio.

The result returned from the SPREAD function is as follows:

```
HS.EXP "A#TempTelephone.C1#[None] = A#Telephone.C1#[None].P#[Year]" (Where  
Period.Number = 1)
```

```
HS.EXP "A#Telephone.C1#[None] = A#TempTelephone P#January *  
E.Globals.A#Profile445.C1#[None].P#Cur /  
E.Globals.A#Profile445.C1#[None].P#[Year]"
```

### Sample Script

The script contains the following information:

- A sample statement written in the calling routine.
- Variables set in the calling routine and passed to the Spread function.
- Variable names in the calling routine set to be the same as the Spread function.

```
Sub Calculate()  
  
Dim Destination  
Dim Source  
Dim Factor  
Dim FactorN  
Dim FactorD  
Dim Temp  
Dim Per  
  
Source = "A#Telephone.C1#[None].P#[Year]"  
Destination = "A#Telephone.C1#[None]"  
Factor = "A#Factor.C1#[None]"  
FactorN = "E#Globals.A#Profile445.C1#[None].P#CUR"  
FactorD = "E#Globals.A#Profile445.C1#[None].P#[Year]"
```

```

Temp = "A#TempTelephone.C1#[None]"
Per = "January"

Call Spread(Destination,Source,Factor,
FactorN,FactorD,Temp,Per)

End Sub

' Beginning of the Spread function

Sub Spread(Destination,Source,Factor,FactorN,FactorD,Temp,Per)

If HS.Period.Number = 1 Then

HS.Exp Temp & " = " & Source

End If

HS.Clear Factor
HS.EXP Factor & " = " & FactorN & " / " & FactorD

HS.Clear Destination
HS.EXP Destination & " = " & Temp & ".P#" & Per & " * " & Factor

End Sub

```

## Units\_Rates

This function calculates the product of two source POVs ( $C = A * B$ ).

### Return Value

No return value.

### Syntax

```
Units_Rates(Destination,Units,Rates)
```

**Table 12-25 Syntax of Units\_Rates Function**

Parameter	Valid Values
Destination	A valid destination POV that is a valid combination of Account, ICP and Custom 1-4 members.



**Table 12-25 (Cont.) Syntax of Units\_Rates Function**

Parameter	Valid Values
Units	A valid source POV that is a valid combination of dimension members.
Rates	A valid source POV.

**Detailed Description**

This function calculates the product of two source POVs ( $C = A * B$ ). As a best practice, it is recommended that variables are set in the calling routine and passed to the Units\_Rates function, which define the Destination, Units and Rates POVs. It is also recommended that the variable names in the calling routine are set to be the same as the Units\_Rates function.

**Example**

The Sales account will return the value of UnitsSold \* Price.

**Table 12-26 Example of Units Sold Function**

Account	Jan2014	Feb2014	Mar2014
A#UnitsSold	1000	2000	5000
A#Price	1.25	1.00	0.50
	1250	2000	2500
Units_Rates("A#UnitsSold",A#Price)			

The result returned from the UNITS\_RATES function is as follows:

```
HS.EXP "A#Sales = A#UnitsSold * A#Price"
```

**Sample Script**

The script contains the following information:

- A sample statement written in the calling routine.
- Variables set in the calling routine and passed to the Units\_Rates function.

- Variable names in the calling routine set to be the same as the Units\_Rates function.

```
Sub Calculate()  
  
Dim Destination  
Dim Units  
Dim Rates  
  
Destination = "A#Sales"  
Units = "A#UnitsSold"  
Rates = "A#Price"  
  
Call Units_Rates(Destination,Units,Rates)  
  
End Sub  
  
' Beginning of the Units_Rates function  
  
Sub Units_Rates(Destination,Units,Rates)  
  
HS.EXP Destination & " = " & Units & " * " & Rates  
  
End Sub
```

## Creating Rules Using Calculation Manager

The Oracle Hyperion Calculation Manager module provides a common user interface to create calculation rules for Oracle Hyperion Financial Management. The graphical flow provides a better understanding of the calculation process and enables you to switch between the graphical view and the VB Script view. Calculation Manager provides a central repository to maintain all calculation rules, and share rules among applications. You can import, export, and print calculation rules, and create custom folders for easy navigation.

### Calculation Manager Security Roles

These roles are available for Oracle Hyperion Calculation Manager access for Oracle Hyperion Financial Management:

- Rules Administrator - can perform any tasks in Calculation Manager for the specified application, such as create, modify and delete rule objects, templates and variables, and validate and deploy any rule sets
- Rules Designer - can create rules objects and modify or delete those objects
- Rules Viewer - can view and validate rules objects

To access Calculation Manager from Oracle Hyperion Enterprise Performance Management Workspace, you must have the Rules Administrator, Rules Designer, or Rules Viewer security role.

To deploy rules, you must have the Rules Administrator security role.

For more information on security roles, see the *Oracle Enterprise Performance Management System User Security Administration Guide*.

### Working with Applications in Calculation Manager

When you create an application, you can load VB script rules, or use Oracle Hyperion Calculation Manager to design and deploy rules to the Oracle Hyperion Financial Management application.

 **Note:**

After you have deployed rules to Calculation Manager, when you attempt to load VB script rules, you will receive a prompt that the Calculation Manager rules will be overwritten. You can either click OK to continue or Cancel.

When you open the Consolidation folder in Calculation Manager, the system displays a list of your applications in alphabetical order.

You can expand the application folder to view Rule Sets, Rules, Formulas, Scripts, and Templates. For information on using Calculation Manager, see the *Oracle Hyperion Calculation Manager Designer's Guide* or online help.

## Migrating Rules to Calculation Manager

If you have existing VB Script rule (.rule) files from a previous release, you can migrate the files into Oracle Hyperion Calculation Manager by importing them. See the "Migrating Rules" section of the *Oracle Hyperion Calculation Manager Designer's Guide*.

## VB Function Support in Function Selector

The Oracle Hyperion Calculation Manager Function Selector supports these VB functions.



### Note:

You can also use other VB functions in the script component, even though they are not available in the UI for selection.

### Array Functions

Function	Description
Array	Returns a variant containing an array
Filter	Returns a zero-based array that contains a subset of a string array based on a filter criteria
Join	Returns a string that consists of a number of substrings in an array
LBound	Returns the smallest subscript for the indicated dimension of an array
Split	Returns a zero-based, one-dimensional array that contains a specified number of substrings
UBound	Returns the largest subscript for the indicated dimension of an array

### Date Time Functions

Function	Description
Date	Returns the current system date
DateAdd	Returns a date to which a specified time interval has been added
DateDiff	Returns the number of intervals between two dates
DatePart	Returns the specified part of a specified date
DateSerial	Returns the date for a specified year, month, and day
Day	Returns a number that represents the day of the month (between 1 and 31, inclusive)
Month	Returns a number that represents the month of the year (between 1 and 12, inclusive)
MonthName	Returns the name of a specified month

Mathematical

Function	Description
Abs	Returns the absolute value of a specified number
Fix	Returns the integer part of a specified number
Int	Returns the integer part of a specified number

String

Function	Description
InStr	Returns the position of the first occurrence of one string within another. The search begins at the first character of the string.
InStrRev	Returns the position of the first occurrence of one string within another. The search begins at the last character of the string.
LCase	Converts a specified string to lowercase
Left	Returns a specified number of characters from the left side of a string
Len	Returns the number of characters in a string
Mid	Returns a specified number of characters from a string
Right	Returns a specified number of characters from the right side of a string
StrComp	Compares two strings and returns a value that represents the result of the comparison
Trim	Removes spaces on both the left and right side of a string
UCase	Converts a specified string to uppercase

## Special VB Script Functions for Financial Management

These special functions were implemented for Oracle Hyperion Financial Management to address array and loop in the Oracle Hyperion Calculation Manager UI.

### Range

Value	Loop Variable	VBScript Generation
@Range(1-50)	i	Dim i(50) I(1)=1 I(2)=2 I(3)=3 .. I(50)=50

Value	Loop Variable	VBScript Generation
@Range(5-10)	i	Dim i(6) l(1)=5 l(2)=6 l(3)=7 l(4)=8 l(5)=9 l(6)=10
@Range(1,3-5,7-9)	i	Dim i(8) l(1)=1 l(2)=3 l(3)=4 l(4)=5 l(5)=7 l(6)=8 l(7)=9

### For / ForStep

Value	Loop Variable	VBScript Generation	Comments
@For(2,10)	Item	For Item=2 to 10	New @ForLoop @For(from, to)
@ForStep(2,10,2)	Item	For Item =2 to 10 step 2	New @ForStep loop function @ForStep(from, to, step) If you need a reverse step, add a negative sign (-) in front of the step, for example: @ForStep(2,10,-2)

### ExitFor

@ExitFor - exiting the loop

The system generates these VB script statements:

For each *element* in group

[*statements*]

### Exit For

*statements*

Next [*element*]

Or

For *counter*=start To end [Step *step*]

[*statements*]

### Exit For

[*statements*]

Next [*counter*]

### **ExitSub**

@ExitSub - exiting the rule

The system generates these VB script statements:

Sub *name* [(*x,y*)]

*statements*]

### **ExitSub**

[*statements*]

EndSub

### **ReDim**

Redimensions one or more dynamic array variables and reallocates their storage space. The optional Preserve keyword can be used to keep the contents of the array intact when it is being redimensioned.

{VarArrayX(5)} = @Redim

{VarArrayXY(5,9)} = @Redim

The system generates these VB statements:

Redim VarArrayX(5)

Redim VarArrayXY(5,9)

### **RedimPreserve**

{VarArrayXY(5)} = @RedimPreserve

Or

{VarArrayXY(5,9)} = @RedimPreserve

Or

{VarArrayXY(5,{*i*})} = @RedimPreserve

The system generates this VB statement:

RedimPreserve VarArrayX(5)

Or

RedimPreserve VarArrayXY(5,9)

Or

RedimPreserve VarArrayXY(5,*i*)

# Managing Intercompany Transactions

## Related Topics

- [Setting Up Intercompany Transactions](#)
- [Managing Reason Codes](#)
- [Monitoring Intercompany Transactions](#)
- [Locking and Unlocking Entities](#)
- [Viewing the Intercompany Transactions Summary](#)

## Setting Up Intercompany Transactions

An intercompany transaction is a transaction between two entities in an organization. Oracle Hyperion Financial Management enables you to track and reconcile intercompany transaction details across accounts and custom dimensions. The Intercompany Transactions module provides an efficient way to identify, report, and reconcile intercompany account differences.

You use the Manage Intercompany Transactions page to process intercompany transactions. For information on processing transactions, see the *Oracle Hyperion Financial Management User's Guide*.

Before you can enter intercompany transactions, you must complete these setup procedures:

- Open periods for the intercompany transactions. See [Opening Intercompany Periods](#).
- Define and load intercompany transaction rules into the application. The HS.SupportsTran function specifies the accounts, scenarios, and custom dimensions in the application that support intercompany transactions. See [Creating Rules Files](#) and [Loading Rules](#).
- Determine matching tolerances. See [Setting Matching Tolerances](#).
- Define reason codes to indicate why a transaction has a MisMatched status. See [Adding Reason Codes](#).
- Enter currency conversion rates. See [Defining Currencies](#).

## Opening Intercompany Periods

Before you can enter, load, or process intercompany transactions, you must first open the period for the transactions. A period can have a status of Unopened, Opened, or Closed. The default status for periods is Unopened. After a period is opened and a transaction has been entered, it can only be changed to Closed. It cannot revert to Unopened.

For each period, you can set the Match/Validate Before Post option and specify matching tolerances that apply to the Auto-Match and Manual Match processes. See [Setting Match/Validate Before Post Option](#) and [Setting Matching Tolerances](#).

To open intercompany periods, you must have the Application Administrator security role.

To open periods:

1. Select **Consolidation**, then **Intercompany**, and then **Periods**.



2. From **Scenario**, select a scenario for the period.
3. From **Year**, select a year for the period.
4. Select each period that you want to open.
5. **Optional:** Enter a Transaction ID Tolerance amount or percentage, Account Tolerance amount, or Manual Tolerance amount for the period.
6. In the **Match/Validate Before Post** column, select an option:
  - If you require the system to check the match status before posting transactions, select **Yes** or **Restrict**.
  - If you do not require the system to check the match status, select **No**.
7. Click **Open Period**, or select **Actions**, and then **Open Period**.

The current status of the period changes to Opened.
8. **Optional:** To save the settings for the period, click **Save Period Settings**, or select **Actions**, and then **Save Period Settings**.

 **Tip:**

To restore the periods to their original status without saving changes, click **Restore**, or select **Actions**, and then **Restore**.

## Setting Matching Tolerances

You can set matching tolerances by period for the Auto-Match and Manual Match processes. You can set amounts for the Account and Manual Matching tolerance. For Transaction ID (TID) tolerance, you can specify an amount, a percentage, or both.

If you enter a percentage for Transaction ID, the system uses the smaller amount between the total of the entity's transaction and the total of the partner's transaction and applies the percentage to the amount, resulting in the tolerance amount.

For example, suppose you have three transactions from Entity A with TID 123, and the total of these transactions is 1000. Partner B with TID 123 has five transactions with a total of 1020. The difference between the entity total and the partner total is 20. However, if you specify a tolerance of 3%, the system calculates 3% of the smaller total, which is 1000 times 3%, resulting in 30. If you compare that to the difference, it would be within the tolerance and the transactions would be considered matched.

If in addition to the percentage, you enter an amount, for example, 15, the system compares the percentage amount with the amount entered and uses the smallest amount as comparison. In this example, the difference between the entity total and the partner total is 20 and the percentage tolerance is 30, but the amount tolerance is 15. This would not be considered within the tolerance and the transactions are not matched.

You can also leave the TID amount and percentage blank. If either has a zero value, or if both are blank, the system matches only transactions that have zero transaction difference.

For Account tolerance and Manual Match tolerance, the matching tolerance is represented in the application currency and the amount entered is represented in the scale factor of the application currency. During the matching process, the system converts each transaction into the application currency and compares the total difference amount to the matching tolerance set for the period. The comparison is done in units.

During the TID/RID matching process, the system does not translate the transaction to the application currency when comparing it to the TID tolerance value.

## Setting Match/Validate Before Post Option

For each period, you can set the Match/Validate Before Post option. The Match/Validate option defines whether the system needs to check the match status of the transactions before the transactions can be posted and defines the types of validation that need to be done before an entity can be locked or a period can be closed.

If you select the Match/Validate Before Post option, you must match transactions or assign a reason code before they can be posted. Before you can close a period or lock an entity, you must post all matched transactions or mismatched transactions with reason codes.

If you select the Restrict option, you must match transactions before you can post them, but you can close periods or lock entities that have unposted transactions.

**Table 14-1 Match/Validate Before Post Option Value Descriptions**

Match/Validate Option Value	Description
No	All transactions can be posted
Yes	<p>The system allows transactions to be posted if either of these conditions are met:</p> <ul style="list-style-type: none"> <li>• Transactions have the Matched status.</li> <li>• Transactions with a MisMatched status contain a valid Reason Code.</li> </ul> <p>Unmatched transactions or mismatched transactions without a reason code cannot be posted.</p> <p>The system also checks that all matched transactions or mismatched transactions with reason codes are posted before the period can be closed or the entity can be locked.</p>
Restrict	<p>If you set the Match/Validate Before Post option to Restrict, the system allows transactions to be posted if either of these conditions are met:</p> <ul style="list-style-type: none"> <li>• Transactions have the Matched status.</li> <li>• Transactions with a MisMatched status contain a valid Reason Code.</li> </ul> <p>Unmatched transactions or mismatched transactions without a reason code cannot be posted.</p> <p>You can close periods or lock entities that have unposted transactions.</p>

## Closing Intercompany Periods

After processing intercompany transactions, you can close the period to prevent modifications to the transactions. If you select the Match/Validate Before Post option, you must post all matched transactions and all mismatched transactions with a reason code. Although the period is locked for future transactions, you can view transactions and run reports for the period.

To close intercompany periods, you must have the Application Administrator security role.

To close a period:

1. Select **Consolidation**, then **Intercompany**, and then **Periods**.
2. From the **Scenario** list, select a scenario for which to close periods.
3. From the **Year** list, select a year in which to close periods.
4. Select the period or periods to close.
5. Click **Close Period**, or select **Actions**, and then **Close Period**.

## Managing Reason Codes

When intercompany transactions are created in the application, they have a default match status of UnMatched. During the Auto-Match process, the match status is updated to Matched or MisMatched.

You can define reason codes to indicate why a transaction has a MisMatched status. For example, this might be due to a missing invoice from the partner entity, or an incorrect amount entered by the partner. After you define the list of valid reason codes for the application, users can select from the list and assign one of the codes when they enter intercompany transactions.

If the Match/Validate Before Post option is selected for the period, you can post transactions with a Matched status, or transactions with a MisMatched status that contain a valid reason code.

You can add, edit, or delete reason codes for an application.

See these procedures:

- [Adding Reason Codes](#)
- [Editing Reason Codes](#)
- [Deleting Reason Codes](#)

## Adding Reason Codes

You can create a list of reason codes for mismatched transactions for an application. Users can then assign one of the reason codes for transactions.

You can manually add reason codes or you can load reason codes during the transaction load process. For information on loading transactions, see the *Oracle Hyperion Financial Management User's Guide*.

To add reason codes:

1. Select **Consolidation**, then **Intercompany**, and then **Reason Codes**.
2. Click **Add Row** or select **Actions**, and then **Add Row**.
3. For **Name**, enter a label for the reason code.

### Note:

The label can contain a maximum of 20 characters. Note that a space is counted as a character. These characters are invalid: ampersand (&), asterisk (\*), backslash (\, comma (,), curly brackets ({}), forward slash (/), hyphen (-), number sign (#), period (.), plus sign (+), and semi-colon (;).

4. **Optional:** For **Description**, enter a description for the reason code.



**Note:**

The description can contain a maximum of 40 characters.

5. Click **Save**, or select **Actions**, and then **Save**.

## Editing Reason Codes

After you create a reason code, you can edit the code description. You cannot edit the reason code label.

To edit reason codes:

1. Select **Consolidation**, then **Intercompany**, and then **Reason Codes**.
2. From the list of reason codes, select the reason code to edit.
3. From the **Description** column, edit the description, and click **OK**.

## Deleting Reason Codes

You can delete reason codes that you no longer need in the list of reason codes for an application.

To delete reason codes:

1. Select **Consolidation**, then **Intercompany**, and then **Reason Codes**.
2. From the list of reason codes, select the reason code or codes to delete.
3. Click **Delete Row** or **Delete All**, or select **Actions**, and then **Delete Row** or **Delete All**.
4. Click **Yes** to delete the reason code.

## Monitoring Intercompany Transactions

You use the Monitor Intercompany Transactions feature to monitor the intercompany transaction matching process. When a large number of intercompany transactions are entered to the system in a period, the matching process can be time-consuming to ensure that all transactions are entered and matched successfully. Since not all transactions are entered at the same time, administrators need to monitor the matching process. The Monitor Intercompany Transactions feature enables you to easily find out which intercompany partners have started their intercompany transactions process.

To monitor Intercompany transactions, you must be assigned the Intercompany Transaction Administrator security role, which enables you to view Process status, Lock status, entity details, intercompany transactions summary, and perform email alerts.

The Intercompany Transactions Monitor page displays a list of intercompany entities with their Process status and Lock status. You can filter the display by Entity, Process status or Lock status. The entities are links to Intercompany Transactions Monitor Detail information. When you click an entity, the system displays the number of posted and unposted transactions by status, such as Matched, Mismatched, or Unmatched.

**Table 14-2 Entity Status for Intercompany Transactions**

Status	Description
Not Started	The entity has no intercompany transactions for the scenario, year, and period in the point of view.
Started	The entity has at least one intercompany transaction for the scenario, year and period in the point of view.
Not Lockable	An entity has a status of Not Lockable if you selected Match/Validate Before Post for the period, but you did not post matched transactions or mismatched transactions with a reason code.

You can click an entity in the list to view the total number of posted and unposted transactions for the entity, categorized by matching status.

For example, if you click the plus (+) sign next to an Entity A to expand it, the page displays the number of posted and unposted transactions by status for Entity A. If you click the value for unposted transactions in the Unmatched column, the system links to the Manage Intercompany Transactions page, with the filtering selected to display the unposted transactions with an Unmatched status for Entity A. You can expand multiple entities simultaneously to view their details.

You can send email alerts for any of the entities in the Intercompany Transactions Monitor page. See the *Oracle Hyperion Financial Management User's Guide*.

To monitor intercompany transactions:

1. Select **Consolidation**, then **Intercompany**, and then **Monitor**.
2. In the Point of View bar, select a scenario, year, and period.
3. From the **Display Options** list, select an option:
  - To view the entity information using the label, select **Label**.
  - To view the entity information using the description, select **Description**.
  - To view the entity information using the label and description, select **Both**.
4. From the **Filters** list, for **Entity**, enter or browse for the entity for which to monitor status.

 **Note:**

If you leave Entity blank, the system returns all entities in the list.

5. **Optional:** To display only active entities, select **Show Active Only**.
6. **Optional:** To filter the transactions list by Process Status, select one or more of these transaction types:
  - **Not Started**
  - **Started**
7. **Optional:** To filter the transactions list by Lock Status, select one or more of these transaction types:
  - **Lockable**

- **Not Lockable**
  - **Locked**
8. **Optional:** To sort the transactions list by **Process** or **Lock** status, click the column heading and select **Ascending** or **Descending**.
  9. Click the plus sign (+) next to an entity or select **Show Details** from the context menu to display the Intercompany Transactions Monitor detail.
  10. From the **Intercompany Transactions Monitor Detail** box, click a value from one of the status columns.  
  
The Manage Intercompany Transactions page is displayed with the selected filter criteria.
  11. **Optional:** To send an email alert for an entity, from the Intercompany Transactions Monitor page, select an entity and click **Email Alert**, or select **Email Alert** from the context menu, or select **Actions**, and then **Email Alert**.

## Locking and Unlocking Entities

You can apply a lock to an entity for a scenario, year, and period to prevent future changes to intercompany transactions for the entity. If the Match/Validate Before Post option is selected for the period, you must post all matched transactions and all mismatched transactions with a reason code before you can lock the entity.



### Note:

The transaction lock status is different from the data lock status. For information on data lock status, see the *Oracle Hyperion Financial Management User's Guide*.

If you are assigned the Intercompany Transaction Administrator security role, you can view Process status, Lock status, entity details, intercompany transactions summary, and perform email alerts.

When the entity is locked, you cannot enter new intercompany transactions. You also cannot delete or change existing transactions. You cannot post or unpost transactions to a locked entity, or update the match status of a transaction that involves a locked entity. Therefore, even if the partner entity is not locked, the partner cannot match transactions because the match status cannot be updated for the entity.

For example, suppose Entity A is locked. You cannot enter any more transactions for Entity A, and no posting or matching can be done to the entity. You can still have intercompany transactions for Entity B with its partner Entity A if Entity B is not locked. However, if you try to match Entity B with Entity A, the process fails because the system cannot update the match status for Entity A.

If you are using submission phases, an entity should not be locked until all phases have Published status.

To lock an entity:

1. Select **Consolidation**, then **Intercompany**, and then **Monitor**.
2. From **Scenario**, select a scenario for the entity.
3. From **Year**, select a year for the entity.
4. From **Period**, select a period for the entity.

5. Select the entity or entities to lock.
6. Click **Lock**, or select **Lock** from the context menu, or select **Actions**, and then **Lock**.

 **Tip:**

To unlock entities, select the entity to unlock, then click **Unlock**, or select **Unlock** from the context menu, or select **Actions**, and then **Unlock**.

To unlock multiple periods, use the **Ctrl** key to highlight the periods, and then select **Unlock**. Note that you cannot unlock multiple periods using the Shift key to highlight them.

## Viewing the Intercompany Transactions Summary

You can view a summary of the status for all intercompany transactions, and if necessary, filter entities to display specific entity

To view the intercompany transactions summary:

1. Select **Consolidation**, then **Intercompany**, and then **Monitor**.
2. Select a Scenario, Year, and Period for the entity.
3. Using the Entity filter, select one or more entities for which to view a summary of transactions.
4. Click **Summary**, or select **Summary** from the context menu, or select **Actions**, and then **Summary**.
5. When you finish viewing the summary, click **OK**.

# Managing Process Management Submission Phases

Process management is the management of the review and approval of financial data. For the purpose of review, data is organized into process units, which are the combination of data for a specific Scenario, Year, Period, Entity, and Value dimension. During the review cycle, you perform actions on process units, such as promote, submit, approve, reject, and publish.

In Process Management, you can divide a data process unit into different submission phases to work with subsets of data. During the review process, you can promote each phase of the process unit rather than the entire process unit. This eliminates the need for additional scenarios to enforce the review process.

To manage Process Management Submission Phases, the UseSubmissionPhase application attribute must be enabled, and you must be an Application Administrator or Review Supervisor.

## Defining Submission Phases

Your review process requirements may vary by period. For example, the monthly close cycle might require a single-phase review process for Balance Sheet and Profit/Loss data in January and February. For a quarterly month such as March, the quarterly closing review process may require multiple phased submission cycles for Balance Sheet and Profit/Loss data and supplemental data.

Your review process requirements may also vary by scenario. For example, the Actual scenario might require only Balance Sheet and Profit/Loss accounts to be submitted for review. For the Budget scenario, all accounts might be required, and for the Forecast scenario, only Profit/Loss accounts and supplemental data might be required.

You can use a maximum of nine submission phases in the review process. For example, in the Actual scenario, you might submit Balance Sheet and Profit/Loss accounts for review in Phase 1, and supplemental data in Phase 2. In the Budget scenario, you might submit Intercompany data in Phase 1, Balance Sheet and Profit/Loss accounts in Phase 2, and supplemental data in Phase 3.

You perform these tasks to set up submission phases:

- Set the application and dimension metadata attributes to use submission phases. See [Defining Application Settings](#).
  - Set the UseSubmissionPhase application attribute to Y. By default, the setting for submission phases is disabled, and you must set this attribute to enable phased submissions in the application.
  - Set SupportSubmissionPhaseForAccounts, SupportSubmissionPhaseForCustom, or SupportSubmissionPhaseForICP attributes as required. You must define which dimensions (Account, Custom, and ICP members) are enabled for process management. For example, if the application needs submission by accounts and not for Custom or Intercompany (ICP) dimensions, you can select the SupportSubmissionPhaseForAccounts attribute. You must enable at least one dimension.



- Assign validation accounts. See [Defining Application Settings](#).  
The validation account is used in Process Control and for data locking. The validation account amount must be zero before data can be promoted, approved, or locked.  
If you do not want to use validation in Process Control or for data locking, leave the validation account blank.  
If you are using phased submissions, you can specify a separate validation account for each submission phase. An application can have up to nine submission phases. For applications that use phased submissions, the Validation Account setting identifies the validation account for phase 1. Validation accounts 2 through 9 identify the remaining phases.
- Define submission groups and assign submission groups to dimension members. See [Setting Up Submission Groups](#).
- Assign submission groups to submission phases. See [Assigning Submission Groups to Phases](#).

## Setting Up Submission Groups

For each application, you must decide which dimension members to include in a submission group. For example, you can define Cash accounts and Investment accounts for Submission Group 1.

In the metadata file, you set this attribute for submission groups: SubmissionGroup=0 to 99 or <blank>.

The default is blank. A <blank> value defaults to the value of 1.

If you set the submission group to zero, the account is not included in the review process.

When multiple dimensions are used for phased submissions, the system determines the cell submission group assignment by the maximum of the group assignments of its dimension members. You should consider all of the group assignments that you need before assigning submission groups.

Example 1:

Account=2

C1=1

C2=2

C3=1

C4=1

ICP=1

The submission group value for the cell is 2 because the maximum submission group number for these dimensions is 2.

**Example 2:**

Account=1

C1=3

C2=2

C3=5

C4=1

ICP=3

The submission group value for the cell is 5 because the maximum submission group number for these dimensions is 5.

## Submission Group and Phase Examples

After you set application and dimension member attributes for phases, and define submission groups for dimension members, you can assign submission groups for accounts to each of the submission phases. The assignment applies only to the scenario that supports Process Management. However, the assignment must be done by Scenario and Period.

These examples show sample submission groups and their assignments to submission phases.

<b>Accounts</b>	<b>Submission Group</b>
HistData	0
Cash	1
Invest	1
ICRec	2
ICPay	2
Liability	3
Equity	3
Revenue	4
Expense	4
SuppData1	5
SuppData2	5
Headcount	6
MiscData	6

<b>C1 (Product)</b>	<b>Submission Group</b>
[None}	1
Golf Balls	7
Tennis Balls	8

C1 (Product)	Submission Group
Soccer Balls	9

Base accounts do not inherit submission groups from parent accounts, and a parent account does not assume any submission group from its children. You must assign a submission group to each account. In this example, the HistData account has a submission group assignment of 0, which means that the account does not require review process.

This example shows submission group assignments by period for the Actual scenario.

**Table 15-1 Example: Submission Group Assignment to Submission Phases**

Period	Submission Phase 1	Submission Phase 2	Submission Phase 3
January (Single Phase)	1, 2, 3, 4	N/A (No supplemental or product data required)	N/A (No supplemental or product data required)
February	1, 2, 3, 4	N/A	N/A
March (Multiple Phases)	2	1, 3, 4, 7, 8, 9	5, 6
April	1, 2, 3, 4	N/A	N/A
May	1, 2, 3, 4	N/A	N/A
June (Multiple Phases)	2	1, 3, 4, 7, 8, 9	5, 6
July	1, 2, 3, 4	N/A	N/A
August	1, 2, 3, 4	N/A	N/A
September (Multiple Phases)	2	1, 3, 4, 7, 8, 9	5, 6
October	1, 2, 3, 4	N/A	N/A
November	1, 2, 3, 4	N/A	N/A
December (Multiple Phases)	2	1, 3, 4, 7, 8, 9	5, 6

### January — Single Phase Assignment

For the January monthly close, in this example, Process Management is required for Submission Phase 1 but not required for Submission Phases 2 and 3. Since this is a short monthly close cycle, a Intercompany, Balance Sheet, and Profit/Loss data (Groups 1, 2, 3, and 4) is submitted in the same submission phase. Supplemental data is not required.

### March — Multiple Phase Assignment

During the March quarterly close, Process Management utilizes data submission across multiple phases.

Submission Phase 1 in this example requires intercompany data, so includes data for ICRec and ICPay accounts; in this example, all accounts with a submission group assignment of 2.

Submission Phase 2 for March contains Balance Sheet and Profit/Loss accounts (Cash, Invest, Liability, Equity, Revenue, and Expense accounts); in this example, all accounts with a submission group assignment of 1, 3, or 4.

Submission Phase 3 for March includes supplemental data, and contains supplemental data accounts such as SuppData1, SuppData2, Headcount, and MiscData; in this example, all accounts with a submission group assignment of 5 and 6.

## Assigning Submission Groups to Phases

You can assign submission groups to each submission phase. The assignment applies only to the scenario that supports process management. If a group is not specified, process management is not applied to the dimensional members within that group.

To assign process management submission groups, you must be an Application Administrator or Review Supervisor.

You assign submission groups to phases by Scenario and Period. You can enter one or more groups for a submission phase and use a comma as a separator for multiple groups assignment (for example, 1, 5, 6, 8, 9). You can specify a range of groups. For example, to assign groups 1, 2, 3, 4, 5, 7 and 8 to a submission phase, you can specify 1-5, 7-8. If you enter one or more groups in a range using commas, when you submit and refresh the data, the system displays the groups that are in a range (for example, 1, 2, 3, 4 displays as 1-4).

Valid groups are 1 to 99. The default for Submission Phase 1 is the keyword ALL to indicate all groups. All groups belong to Submission Phase 1 until you change their assignment.

You cannot assign the same group to multiple phases in the same period. For example, you cannot specify Groups 2 through 5 for Phase 1, and Groups 3 and 8 for Phase 2, because Group 3 cannot be assigned to both Phase 1 and Phase 2. A submission group can only be assigned to one phase in the same period. An error message displays if you try to assign a submission group to a phase with one already assigned. If you move all groups out of a submission phase into a different submission phase, the original phase is cleared entirely from the system for the specified scenario and period.

You can skip a submission phase assignment. For example, you can assign groups to submission Phases 1 and 3 without assigning groups to Phase 2. Any groups that are not assigned to a submission phase are not considered part of the review process. Those dimensional members are available to all users with the appropriate security class access without the need to check for review level security. Unassigned cells do not need to be started for process management before you can enter data.

To assign submission groups for submission phases:

1. Select **Consolidation**, then **Data**, and then **Submission Phases**.

By default, all phases are displayed. From the Options pane on the right, you can clear phases that you want to hide.

2. To change the scenario, click the Scenario dimension in the POV, select a scenario, and click **OK**.
3. In each phase column, enter the groups for that submission phase and press **Enter**.
  - To enter multiple groups, use a comma as a separator.
  - To specify a range of groups, use a dash as a separator.
  - To indicate all groups, specify ALL.
4. Click **Submit** or select **Actions**, and then **Submit** to save the data.
5. click **Refresh**, or select **Actions**, and then **Refresh** to refresh the database.

## Viewing Unassigned Submission Groups

In Process Control, you can view submission groups that are not assigned to any phase. Submission groups that are not assigned a submission phase are not part of the review

process. Viewing unassigned groups enables you to check if you have accidentally omitted groups from the review process.

The system displays groups assigned to dimension members that are not assigned to any submission phase, and groups assigned to a submission phase that are not assigned to any dimension member. If a group has not been assigned to either a dimension member or phase, it is not displayed.

For example, an application has submission groups 1-10 assigned:

Period	Submission Phase 1	Submission Phase 2	Submission Phase 3
January	1,2,3,4	5,7	8,9
February	1,2,3,4	5-8	N/A
March	2	1,3,4	5,6
April	1,2,3,4	5,6	8
May	1,2,3,4	5-8	N/A
June	2	1,3,4	5,6
July	1,2,3,4	N/A	N/A
August	1,2,3,4	N/A	N/A
September	2	1,3,4	5,6
October	1,2,3,4	N/A	N/A
November	1,2,3,4	NA	N/A
December	2	1,3,4	5,6

If you select the option to display unassigned groups, for January in the previous example, these groups are shown as unassigned:

Period	Unassigned Group
January	6,10

To view unassigned submission groups:

1. Select **Consolidation**, then **Data**, and then **Submission Phases**.  
The Unassigned Groups information is displayed in the right pane.
2. Select a row for the period, and from the **Unassigned Groups** pane, click **Refresh**.

## Managing Email Alerting

You can use email alerting for intercompany transactions and during the process management review process. Email alerts help highlight a key event or data change in the system. For example, you can send an email alert that an intercompany transaction is mismatched and needs to be matched, or that a process unit is ready for the next promotion level.

Email alerts are sent using standard Simple Mail Transfer Protocol (SMTP), so you can use alerts with any email system that works with Internet email. To use alerts, you must configure email settings and specify the SMTP server name when you run the Financial Management configuration utility. See the *Oracle Enterprise Performance Management System Installation and Configuration Guide*.

The alert process uses the email addresses that are stored in your authentication files, such as LDAP, MSAD, or Native Directory.

Before you can send or receive email alerts, you must have set up user and data security rights in the application. The security class assigned to the scenario and entity for the alert must support email alerts, and users must be assigned a security role to receive email alerts. For information on setting security roles, see the *Oracle Enterprise Performance Management System User Security Administration Guide*.

## Setting Up Process Management Alerting

You can set up the Process Control module to generate email alerts based on a change of status in process control. You can set up alerts for these actions: First Pass, Review levels 1 through 10, Submitted, Approved, or Published.

Email alerts are not generated when the process unit is at the Not Started level or for the Sign Off action.

To set up process management email alerts:

1. In the metadata file, for the scenario in the process unit, set the SupportsProcessManagement metadata attribute to A to allow alerts.

 **Note:**

When you enable this attribute, the scenario generates email alerts during the review process for the users that have the security rights to receive them.

2. Assign the user to the Receive Email Alerts for Process Control security role.
3. Assign the user ALL or PROMOTE access to the security classes assigned to the scenario and entity in the process unit and add an alert for each security class.
4. In the Security Access section for the security class, set the Support Alert option to Y for Yes to enable alerts. For example: [Default];User1@NativeDirectory;All;Y.

Users who meet all criteria receive email alerts.

**Table 16-1 Process Management User Roles and Alert Notification**

Process Unit Level Before or After Action	Process Management User Roles Notified
First Pass	Users with ALL or PROMOTE access to the entity are notified.
Review Levels 1 through 10	Reviewer at that Review Level and Submitter roles are notified. For example, for Review Level 1, Reviewer 1 and Submitter roles are notified.
Submitted	Review Supervisor role is notified. Only users with this role can approve the submitted process unit.
Approved	Reviewer 1 to Reviewer 10 and Submitter roles are notified.
Published	Users with ALL, READ, or PROMOTE access to the entity are notified.

When a process review action is performed, the system automatically generates email alerts to the appropriate users, according to the security rights that have been set up. The user that performed the action is also notified with a confirmation email.



**Note:**

Users with the Application Administrator role do not receive email alerts. For a user with the Application Administrator role to receive email alerts, set up the administrator as a separate user and assign the role to receive alerts.

## Setting Up Intercompany Alerting

You can generate intercompany transaction email alerts for users who are assigned the security rights to receive them.

For information on setting security roles, see the *Oracle Enterprise Performance Management System User Security Administration Guide*.

To set up intercompany transaction email alerts:

1. Assign the user to the Receive Email Alerts for Intercompany security role.
2. In the Security Access section for the security class, set the Support Alert option to Y for Yes to enable alerts. For example: [Default];User1@NativeDirectory;All;Y.

Users who meet all criteria receive email alerts from the Intercompany Transactions or Intercompany Partner Matching Report modules.

For information on generating email alerts in intercompany transactions, see the *Oracle Hyperion Financial Management User's Guide*.

# A

## Configuration Settings

The Settings module enables you to view and modify Oracle Hyperion Financial Management configuration settings. By default, the Settings table is populated with the recommended settings at the time of installation.

You can change setting values if required. For example, you might need to change values based on memory usage or to improve performance. For considerations on updating these values, see [Optimizing Performance](#).

You can override settings for a specific application. To change a system-level value, you must have the Financial Management Administrator security role. To change an application level value, you must have both the Financial Management Administrator and the Application Administrator security roles.

You can sort the settings list, search for settings, and export them to an Excel worksheet. You can delete setting overrides, but you cannot delete settings created by the system.

You can also view runtime values for specific application settings. See [Viewing Effective Settings](#).

See these topics:

- [Available Configuration Settings](#)
- [Changing Configuration Settings](#)
- [Overriding Values](#)
- [Changing the Settings Table Display](#)
- [Searching for Settings](#)
- [Viewing Effective Settings](#)
- [Exporting Settings](#)
- [Deleting Settings](#)



# Available Configuration Settings

**Table A-1 Configuration Settings**

Setting	Description	Values
AllowOverlappingConsolidationOverride	Whether to allow users to run overlapping consolidation processes.	0 - Do not allow users to run overlapping consolidations. A user cannot start a new consolidation process if there is one currently running and overlapping. 1 - Allow users to run overlapping consolidations. If another consolidation process is currently running and overlapping, the system displays a warning message asking whether to start a new consolidation. Default value: 1
AutoClearDeadTasks	Whether to automatically clear completed tasks from the system	0 - Leave completed tasks in the system 1 - Clear completed tasks from the system Default value: 1
AutoClearEAFlatfileTasksAfterSeconds	Number of seconds for the system to wait before automatically clearing an Extract Data task from Running Tasks and deleting the file from the Application Server	Range: 600 seconds (10 minutes) - 864,000 seconds (10 days) Default value: 86,400 seconds
BRCRowFetchSize	Row fetch size for every iteration of the table. Used only for LCM application snapshot export.	Range: 1000 to 100000 rows Default value: 5000
BRCThreadPoolCount	Thread pool size – Number of threads used for processing the LCM application snapshot import and export	Range: 10 to 100 threads Default value: 15

**Table A-1 (Cont.) Configuration Settings**

Setting	Description	Values
ConcurrentCalculationWaitTimeInSecond	<p>If the maximum number of concurrent calculations are running and a user launches a new calculation, the system will wait until the timeout and log a message in the system log about the timeout with the POV details.</p> <p>If the maximum number of concurrent calculations are running and ConcurrentCalculationWaitTimeInSecond = -1, and a user launches a new calculation, then the system will wait infinitely until one of the calculations finishes.</p> <p>This setting can be changed by an administrator per application per server, but it cannot be set lower than 15 seconds.</p>	Default value: 30 seconds
ConsolidationMultiThreadingScheme	<p>Used for multiple settings related to how the consolidation process is run. Each setting can be enabled or disabled without affecting the other settings. You can choose more than one option and set the parameter to the sum of the chosen options. For example, to use options 2 and 4, set the value to 6. A value of 0 represents the legacy consolidation behavior with limited multi-threading where none of the three options are enabled. This setting can be overridden at the server and/or application level.</p>	<p>Valid values:</p> <p>1 - Allows the consolidation process to calculate and translate all children entities up to the [Parent] value before the numbers are consolidated into the parent entity.</p> <p>2 - Allows the consolidation process to multi-thread parent-level entities, greatly improving the performance on large consolidations.</p> <p>4 - Skips the initial calculation on all base-level entities. Those entities are calculated as needed when processing their parent entities.</p> <p>8 - Allows the consolidation process to consolidate only impacted children from the last failure during Consolidate All and Consolidate All with Data</p> <p>Default value: 2</p>
DataSize	Fetch size for table components	Range: 25 - 2,000 Default value: 500
DefaultAdminPage	Default opening page for Consolidation Administration	<p>Valid values: Any of these pages: System Messages, Applications, System Settings, System Users, System Manage, Task Audit, Data Audit, Manage Task Flows, Task Flow Status, Configure DSN</p> <p>Default value: System Messages</p>

**Table A-1 (Cont.) Configuration Settings**

Setting	Description	Values
DefaultColFetchSize	Default column fetch size on Data forms and Data grids	Range: 25 - 2,000 Default value: 50
DefaultRowFetchSize	Default row fetch size on Data forms and Data grids	Range: 25 - 2,000 Default value: 250
DSSStartupOption	Determines when applications are started and shut down. This setting can be set globally or overridden at the application and server level or both.	0 - Application is started on demand, when the first user tries to open the application. Application shuts down after the last user logs off the application. This setting is useful to keep applications up only when necessary. Applications used for archiving do not need to be up all the time. 1 - Application is started on the Java Server startup. Application continues to stay up even after the last user logs off the system. This setting is useful to keep the application up all the time so users do not have long wait times when opening applications. 2 - Application is started on demand. Application continues to stay up even after the last user logs off the system. Default value: 0
EnableRulesLogging	Whether to enable rules logging. Used for debugging purposes to isolate rules issues.	0 - Disable logging 1 - Enable logging Default value: 0
EnableRulesProfiling	Enable rules profiling. You can override the value at Application level or globally.	0 - Disable rules profiling 1 - Enable rules profiling Default value: 0
EnableRunningTasksMaskUserNames	Whether to mask user names in Running Tasks when viewed by users other than administrators	0 - Disable masking of user names 1 - Enable masking of user names Default value: 0

**Table A-1 (Cont.) Configuration Settings**

Setting	Description	Values
IcmSystemReportTranslationMode	<p>When running an Intercompany Matching System Report that requires currency translations, translated values are written to the database by default. This can lead to longer running reports and a significant increase in database size. This setting allows for control over how translated values are handled.</p> <p>Translation mode = 0 (perform sub translate and store data)</p> <p>Translation mode = 1 (perform sub translate and do not store data)</p> <p>Translation mode = 2 (perform on-the-fly translation using the default translation, do not perform sub translate, and do not store data)</p>	<p>0 - Use existing translation logic. Uses Sub Translate in rules and commits changes to the database.</p> <p>1 - On the fly, use rules. Uses Sub Translate in rules, but no translated values are written to the database. Note: Parent entities in CN status are also translated.</p> <p>2 - On the fly, use default translation. Executes only default translation (no Sub Translate). No changes are written to the database. Note: Parent entities in CN status are not translated.</p> <p>Default value: 0</p>
MaxDataCacheSizeInMB	Maximum memory in MB that the application server allocates to store the cell	<p>Range: The value should be greater than or equal to 500.</p> <p>Default value: 4,500</p>
MaxFileSelectionOnLoad	Maximum number of files that can be selected in multiple file selection control	<p>Value must be greater than or equal to 1.</p> <p>Default value: 10</p>
MaxNumConcurrentCalculations	Maximum number of concurrent calculations allowed per server per application. This setting can be changed by the administrator per application per server.	Default value: 8
MaxNumConcurrentConsolidations	Maximum number of concurrent consolidations allowed per server per application. Any consolidations performed above the set value are queued as Scheduled Consolidations. This setting can be set globally or overridden at the server of application level.	<p>Range: No hard limit on range. Value must be a positive number.</p> <p>Default value: 8</p>
MaxNumConcurrentConsolidationsPerAction	Maximum number of concurrent consolidations allowed per user action. Any consolidations performed above the set value are queued as Scheduled Consolidations.	Default value: 8
MaxNumCubesInRAM	Maximum number of subcubes for FreeLRU algorithm	<p>Range: The value should be greater than or equal to 100.</p> <p>Default value: 60,000</p>

**Table A-1 (Cont.) Configuration Settings**

Setting	Description	Values
MaxNumDataRecordsInRAM	Maximum number of data records for the system to store in RAM	Range: No hard limit on range. Value must be a positive number. Default value: 30,000,000
MaxNumRetriesOfBaseLevelCalculation	<p>Maximum times the consolidation process retries the calculation on base-level entities.</p> <p>This setting allows you to repeat the calculation of base-level entities multiple times during a consolidation process up to the specified amount of retries, or until the status of those entities become OK.</p> <p>It is used if the HS.ImpactStatus function in a rule running on one base-level entity impacts another entity that was already calculated, and the top entity status is not OK.</p> <p>Example: California and Connecticut are base-level entities that roll up to the USA parent entity. During the calculation of USA, if you use HS.ImpactStatus in SubCalculate, the Connecticut entity is impacted. After running consolidation, the status for the USA parent entity is not OK due to changes in the Connecticut descendant. If this setting is set to 2, the system retries the calculation process twice. On the second calculation try, California is OK, Connecticut is calculated and rolled up to USA, and the USA entity status is OK. If there are 3 or 4 levels of dependency, you can adjust the setting to retry as many times as needed.</p>	Range: 1-4 Default value: 0
MinDataCacheSizeInMB	Minimum data cache size in MB. By setting this value at a higher number, you can reduce the number of Data Cache growth attempts by the system and reduce memory fragmentation. Typically data cache is increased on a per-need basis, and increases 25 MB maximum at a time.	Range: 100-5,000 Default value: 2,250

**Table A-1 (Cont.) Configuration Settings**

Setting	Description	Values
NumConsolidationThreads	Maximum consolidation threads allowed per consolidation. Reducing the value limits system use of resources, causing slower consolidation performance. There should be the same number of CPU cores available for Oracle Hyperion Financial Management on the system.	Range: No hard limit on range. Value should be greater than or equal to 1. Default value: 8
NumCubesLoadedBeforeCheckingLRU	The number of cubes that must be loaded in RAM to trigger the FreeLRU algorithm. FreeLRU is triggered when either of the two conditions is met: NumMinutesBeforeCheckingLRU is met, or NumCubesLoadedBeforeCheckingLRU is reached.	Range: No hard limit on range. Value must be a positive number. Default value: 100
NumDataLoadsAllowed	Maximum concurrent data load tasks allowed per server, per application. This setting can be set globally or overridden at the server or application level.	Range: No hard limit on range. Value must be a positive number. Default value: 8
NumEAExportsAllowed	Maximum concurrent Extract Data tasks allowed per server, per application. This setting can be set globally or overridden at the server or application level.	Range: No hard limit on range. Value must be a positive number. Default value: 8
NumEAThreads	Number of threads per data extract	Range: No hard limit on range. Value must be a positive number. Default value: 8
NumMinutesBeforeCheckingLRU	Interval in minutes for checking limits of FreeLRU algorithm	The value should be greater than or equal to 1. Default value: 15
NumReportsAllowed	Maximum concurrent report tasks allowed per report type, per server, per application.	Default value: 8
NumThreadsToUseWhenUpdatingCalcStatusSystemWasChanged	Number of threads to use when updating Calculation Status after a metadata load	The value should be greater than or equal to 1. Default value: 16
SessionManagerTimeoutInMS	Timeout limit used by session manager in Financial Management, in milliseconds	The value should be greater than or equal to 60,000. Default value: 1,200,000
OverrideUserFetchSizeWhenOpening	Whether to override the fetch sizes for all users on data forms and data grids	0 - Do not override fetch sizes for all users 1 - Override fetch sizes for all users Default value: 0

**Table A-1 (Cont.) Configuration Settings**

Setting	Description	Values
WebformDoCalculateOnSubmit	Whether to automatically calculate after submitting the data when user clicks Submit Data on a Data form.	0 - Do not automatically calculate 1 - Automatically calculate Default value: 0

## Changing Configuration Settings

To change a setting, you must have Administrator security rights. For system-owned settings, you cannot change any other column than the value.

You can use the Notes column to enter comments; for example, why the value was changed. You can view any existing notes for settings by hovering over the Notes icon for a row.

When a setting is modified, the Settings table is updated to display the name of the user who modified it and the date and time that it was modified.

Settings displayed in blue indicate that a user-modifiable setting doesn't match the default value, and the value has been updated.



### Note:

Using Unicode UTF-8 for worldwide language support is not recommended for Oracle Hyperion Financial Management.

To change a configuration setting:

1. Select **Navigate**, then **Administer**, and then **Consolidation Administration**.
2. From **Admin Tasks**, expand **System**, and then click **Settings**.
3. From the Settings table, change the value for a setting.



### Tip:

To reset to the original values, click **Reset**, and from the confirmation prompt, click **Yes**.

4. **Optional:** To enter a comment, click the **Notes** icon for the row, enter a comment, and then click **OK**.
5. Click **Save**.
6. Click **Refresh** to display the changes.

## Overriding Values

You can override the default value for a setting for a server and application. If a setting is Global, you cannot override it.

To copy a setting:

1. From the Settings table, select a setting to copy.
2. Click **Override**, or select **Actions**, and then **Override**, or right-click a setting and select **Override**.
3. From the Override dialog box, change the setting value, and then select the Server and Application to which the setting applies.
4. **Optional:** To enter a comment, click the **Notes** icon for the row, enter a comment, and then click **OK**.
5. Click **Apply and Close** to apply the new value.

The new setting that you created is displayed in bold and selected in the table.

**Optional:** To override another setting, click **Apply and Override Another**.

6. Click **Save**.

## Changing the Settings Table Display

You can sort any of the setting columns, and you can reorder columns.

To select columns to display:

1. From the **Settings** list, click **View**, then **Columns**, and then select the columns to display, or select **Show All**.
2. **Optional:** To show or hide columns, select **View**, then **Columns**, then **Manage Columns**, and then use the arrow keys to move columns to the Hidden or Visible column lists.
3. **Optional:** To change the column sort order, click the header icons and then select **Sort Ascending** or **Sort Descending**.

## Searching for Settings

You can search for settings based on selected criteria. From the Manage Settings tab, you can search on the setting name, the server and application to which the setting applies, the date it was updated, and the user who created or updated it.

The Effective Settings tab enables you to see what values are used by the server at runtime for an application. You can search for settings by name, server, and application.

To search for settings:

1. From the **Search** fields, select or enter criteria for which to search.
2. Select **All** to match all criteria, or select **Any** to match any selected criteria.
3. Click **Search**.

### Tip:

To reset the Search settings, click the **Reset** button next to Search.



## Viewing Effective Settings

From the Effective Settings tab, you can view runtime values for application settings.

To view effective settings:

1. From **Manage Settings**, select a server name and application name.
2. Click **Effective Settings**, or select **Actions**, and then **Effective Settings**.

The effective settings are based on the selected application and server. Default values that are overridden are displayed in blue.

3. To return to the main settings list, click **Manage Settings**, or select **Actions**, and then **Manage Settings**.

## Exporting Settings

You can export the list of settings and save them to an Excel worksheet.

To export a setting:

1. From Manage Settings, click **Export**, or select **Actions**, and then **Export**.  
The rows based on the current search are exported.
2. Follow the download instructions to download the file as an Excel worksheet.

## Deleting Settings

You can delete settings that are overridden. You cannot delete system-created settings.

To delete a setting:

1. From the Settings table, select the setting row that you want to delete.
2. Click **Delete Selected**, or select **Actions**, and then **Delete Selected**, or right-click a setting and select **Delete Selected**.
3. From the confirmation prompt, click **Yes**.

# B

## Optimizing Performance

This chapter assumes familiarity with the Oracle Hyperion Financial Management application, database administration, and general operating system concepts. Actual implementations and environments will vary widely based on business requirements, Financial Management data set, network topology, and hardware usage. Therefore, you must consider how to adapt these guidelines to your own implementations. All test results and performance numbers are only intended as examples to illustrate tuning concepts.

### **Caution:**

Improper settings and configurations may prevent Financial Management from working.

Before implementing any of the tuning settings, it is recommended to carry out end-to-end performance testing to obtain baseline performance data for the default configurations, make incremental changes to the tuning settings and then collect performance data. It should be verified that configuration changes improve, not impair, system performance.

For supported platforms and components, refer to the Oracle Enterprise Performance Management System Supported Platform Matrix: [Oracle EPM Supported Platform Matrix](#).

## Performance Overview

### Introduction to Oracle Hyperion EPM System Performance

To maximize Oracle Hyperion EPM System performance, you need to monitor, analyze, and tune all the components. This section describes the tools that you can use to monitor performance and the techniques for optimizing the performance of Oracle Hyperion Financial Management.

## Common EPM Installation Directory References

This chapter uses the following directory references:

### **Middleware Home**

A Middleware home consists of the Oracle WebLogic Server home, and, optionally, one or more Oracle homes, including EPM Oracle home. The default installation directory is Oracle\Middleware. The Middleware home location is referred to as *MIDDLEWARE\_HOME* throughout this chapter.

### **EPM Oracle Home**

An Oracle home contains installed files necessary to host a specific product, and resides within the directory structure of the Middleware home. The default EPM Oracle home location is *MIDDLEWARE\_HOME\EPMSys11R1*. The EPM Oracle home location is referred to as *EPM\_ORACLE\_HOME* throughout this chapter.

### EPM Oracle Instance

Additionally, during configuration, some products deploy components to the EPM Oracle Instance defined during configuration. The default location for the EPM Oracle instance is `MIDDLEWARE_HOME\user_projects\epmsystem1`. The EPM Oracle instance location is referred to as `EPM_ORACLE_INSTANCE` throughout this chapter.

## Financial Management Records and Subcubes

A **record** in Oracle Hyperion Financial Management holds the data for all base periods for a given intersection of dimension members.

A **subcube** consists of all the cells that share the same members of the following dimensions:

- Scenario
- Year
- Entity
- Value

There are two types of subcubes: currency subcubes and node subcubes. These types of subcubes differ in how they use Entity and Value dimension members:

- A currency subcube contains cells that share applicable non-node Value dimension members. For currency subcubes, the parent of the Entity member is irrelevant. The applicable non-node Value dimension members are as follows:
  - Members for user-defined currencies. There is one triplet of Value dimension members for each user-defined currency. For example, if an application contains a currency named USD, the currency's triplet of Value dimension members will be USD, USD Adjs, and USD Total. The triplet is still considered as three separate subcubes.
  - The triplet that points to the entity's default currency. This triplet consists of the <Entity Currency>, <Entity Curr Adjs>, and <Entity Curr Total> Value members.
  - [None] Value member.

#### Note:

The non-node Value dimension members that point to parent entities' default currencies - <Parent Currency>, <Parent Curr Adjs>, and <Parent Curr Total> - are irrelevant to currency subcubes.

- A node subcube contains cells that share a common node Value dimension member. For node subcubes, both parent and child Entity members must be specified. The node Value dimension members are as follows:
  - [Contribution Total]
  - [Contribution Adjs]
  - [Contribution]
  - [Elimination]
  - [Proportion]
  - [Parent Total]
  - [Parent Adjs]
  - [Parent]

Because the subcube is a natural unit of data for the purposes of consolidation, data movement and processing are carried out on a subcube basis in many places in Financial Management.

## Tuning Recommendations for Financial Management

Performance tuning Financial Management is an iterative process.



### Note:

Tuning must be done for a particular production workload. Tuning can be conducted when workload is generated by load generation tools like Oracle Application Testing Suite (OATS).

This section discusses several areas that provide a quick start for performance tuning including:

- Tune Operating Systems parameters
- Tune HTTP Server parameters
- Tune Oracle Database parameters



### Note:

While this list is a useful tool in starting your performance tuning, it is not meant to be a comprehensive list of areas to tune. You must monitor and track specific performance issues within your implementation to understand where tuning can improve performance.

## Diagnosing Performance Problems

When a performance issue arises, it is critical to first determine the cause prior to taking any corrective action. Oracle does not recommend changing performance-related parameter settings or taking other actions until an extensive analysis of the problem has been performed.

## Using Monitoring Tools

Oracle strongly recommends using monitoring tools to collect performance data as part of the diagnostic process. Monitoring the Oracle Hyperion Financial Management application server, web servers, database server(s) and network layers provide useful performance data.

On a Linux environment, tools such as OS Watcher Black Box can be used to monitor the operating system and the Financial Management processes.

On a Windows environment, a tool such as Microsoft Performance Monitoring can be used to monitor the Financial Management application process performance. Steps for configuring Performance Monitor to gather the counters specific for Financial Management applications can be found here:

[Configuring Performance Monitor](#)

Performance counters to monitor include:

Subsystem	Counter	Guidance
Memory	Memory: Free System Page Table Entries	Warning when Free System Page Table Entries is less than 8,000. Critical when Free System Page Table Entries is less than 5,000.
Memory	Memory: Available Mbytes	Should be no lower than 20% to 25% of installed physical memory. In these cases, carefully monitor Paging activity.
Memory	Memory: Page Reads/sec	This counter should be below 1,000 at all times.
Processor	Processor: % Processor Time: _Total	Total processor utilization should be lower than about 70% to 80%.
Processor	Processor: % Processor Time: (N)	Each processor instance should be lower than about 70% to 80% utilization.
Disk	Physical Disk: Avg. Disk sec/Transfer	Must be lower than about 25 ms. General rule: When Avg. Disk Seconds/Transfer (the disk latency counter) is significantly greater than 25 ms, the disk subsystem is unhealthy and is a bottleneck. Please note that this counter does not tell you how to fix the problem; it only indicates there is a problem.
Disk	PhysicalDisk: Average Disk Queue Length	The average should be less than the number of spindles of the disk. If a SAN is being used, ignore this counter and concentrate on the latency counters: PhysicalDisk\Average Disk sec/Read and PhysicalDisk\Average Disk sec/Write.
Disk	PhysicalDisk: Average Disk sec/Read	The average value should be below 20 ms. Spikes (maximum values) should not be higher than 50 ms.
Disk	PhysicalDisk: Average Disk sec/Write	The average value should be below 20 ms. Spikes (maximum values) should not be higher than 50 ms.
Network	Network Interface: Bytes Total/sec	For a 100-Mbps network interface card (NIC), it should be below 6–7 MB/sec. For a 1000-Mbps NIC, it should be below 60–70 MB/sec.
Network	Network Interface: Packets Outbound Errors	It should be zero (0) at all times.

Process	Counters / Guidance
Oracle Hyperion Financial Management: Instance: 1 JHsxServer 2. XFMDDataSource	<p>Process: % Processor Time – Process processor utilization should be lower than 90%.</p> <p>Process: Private Bytes - reports bytes allocated exclusively for a specific process; its value tends to rise for a leaking process.</p> <p>Process: Working Set - reports the shared and private bytes allocated to a process; its value tends to rise for a leaking process.</p> <p>Process: Page Faults /sec - reports the total number of faults (hard and soft faults) caused by a process; its value tends to rise for a leaking process.</p> <p>Process: Page File Bytes - reports the size of the paging file; its value tends to rise during a memory leak.</p> <p>Process: Handle Count - reports the number of handles that an application opened for objects creates. Handles are used by programs to identify resources that they must access. The value of this counter tends to rise during a memory leak.</p> <p>Process: Virtual Bytes - The current size, in bytes, of the virtual address space the process is using. The virtual byte of the perfmon process grows at a quick rate and never releases any memory indicating memory leak in application.</p> <p>Process: Virtual Bytes Peak - The maximum size, in bytes, of virtual address space the process has used at any one time. The virtual byte of the perfmon process grows at a quick rate and never releases any memory indicating memory leak in application.</p> <p>Process: Pool Nonpaged Bytes - The size, in bytes, of the Nonpaged pool, an area of system memory (physical memory).</p>

## Using Remote Diagnostic Agent (RDA)

Remote Diagnostic Agent (RDA) is a set of command-line diagnostic scripts that are executed by an engine written in the Perl programming language. The data gathered provides a comprehensive picture of the environment that aids in problem diagnosis.

Running RDA can be particularly helpful in determining the size of the subcube(s) in your Oracle Hyperion Financial Management application(s). RDA is available through the My Oracle Support website. To get started, see this Knowledge Base article:

[Document 1100612.1](#)

## Using a Reference Application

A reference application is an application used to diagnose performance issues. A reference application runs a set number of tasks that can be compared with internal timings to help determine if your environment was properly tuned. Running a reference application of some type can help identify problems. However, there is no one application that can manifest all the performance characteristics of Oracle Hyperion Financial Management. It is very common to

see good performance on one application, and poor performance on another. The parameters involved include data volume, record distribution per subcube, entity structure, number of currencies, and so on.

## Tuning Operating Systems Parameters

### Tuning Windows Parameters

For Windows platforms, the default TCP/IP settings are usually sufficient. However, under heavy loads it may be necessary to adjust the *MaxUserPort* and *TcpTimedWaitDelay*. These parameters determine the availability of user ports requested by an application.

Parameter	Default Value	Suggested Value
<p><b>TcpTimedWaitDelay</b></p> <p>This parameter controls the amount of time the OS waits to reclaim a port after an application closes a TCP connection. It has a default value of 4 minutes. During a heavy users load, these limits may be exceeded resulting in an address- in-use connection exception.</p> <p>Set this parameter in the Windows registry at the following location:            HKLM\System\CurrentControlSet\Services\Tcpip\Parameters            Value Name: TcpTimedWaitDelay            Value Type: DWORD            Data: 30 (decimal)</p>	240	30
<p><b>MaxUserPort</b></p> <p>Value Name: MaxUserPort            Value Type: DWORD            Data: 65534 (decimal)</p> <p>Beginning with Windows2008 it is necessary to execute the following command in order to set the mentioned number of ports:</p> <pre>netsh int ipv4 set dynamicportrange protocol=tcp startport=10100 numberofports=55433 store=persistent</pre> <p>The given command is based on the following articles at <a href="http://support.microsoft.com">support.microsoft.com</a>:            Dynamic Port Range            Windows Settings</p>	5000	65534

## Tuning the Web Server

Key tuning steps for the Oracle Hyperion Financial Management web server and other EPM components are included in the EPM System Configurator, which is installed with the first EPM System product installed on a computer and is used to configure all products installed on the computer. For more information, see the *Oracle Enterprise Performance Management System Installation and Configuration Guide*.

### Tuning HFM Web

#### HFM Web Timeout Parameters

EPM Config files	Default Value	Suggested Value
<b>Web Server Plug-ins with Weblogic</b> The following settings apply when using Oracle HTTP server or IIS to proxy requests to Weblogic. Choice of which web server type is used is set from EPM System Configurator. Parameters are case-sensitive and must be manually added.		
<b>WLIOTimeoutSecs</b> Defines the amount of time the plug-in waits for a response to a request from the WebLogic Server.	7,200	7,200
<b>WLSocketTimeoutSecs</b> Set the timeout for the socket while connecting, in seconds	600	600
<b>(OHS Web Server Plug-In)</b> EPM_ORACLE_INSTANCE\httpConfig\ohs\config\OHS\ohs_component\mod_wl_ohs.conf Applies to hfmadf instance: example: <pre> &lt;LocationMatch ^/hfmadf&gt; SetHandler weblogic-handler WeblogicCluster&lt;WLCluster&gt;:&lt;port&gt; WLIOTimeoutSecs 7200 Idempotent OFF WLSocketTimeoutSecs 600 &lt;/LocationMatch&gt; </pre>		

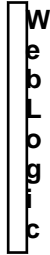


EPM Config files	Default Value	Suggested Value
<b>(IIS Web Server Plug-In)</b>		
EPM_ORACLE_INSTANCE\httpConfig\ VirtualHosts\hfmadf\iisproxy.ini		
WIForwardPath=/hfmadf		
PathTrim=/		
WebLogicHost=<host>		
WebLogicPort=<port>		
KeepAliveEnabled=true		
KeepAliveSecs=20		
WLIOTimeoutSecs=3600		
Idempotent=OFF		
WLSocketTimeoutSecs=750		
<b>Weblogic</b>		
<b>Stuck Threads (Weblogic Admin console)</b>		
Long-running tasks like Metadata loads can appear to be stuck when HFM application process is busy processing the load file. Increase the Stuck Thread Time when these issues occur.		
Change the following setting from the Weblogic console.		
Select <b>Lock and Edit</b> .		
Select Servers and click on HFMWeb(N).		
Select the Tuning tab.		
Change <b>Stuck Thread Max Time</b> to 1200.		
Change <b>Stuck Thread Timer Interval</b> to 1200.		
Select <b>Activate Changes</b> .		
The HFMWeb0 server will need to be restarted.		
Where HFMWeb(N) is HFMWeb0, HFMWeb1, etc., depending on how many HFM web servers are deployed and which server you are updating.		
Stuck Thread Max Time	600	1200
Stuck Thread Timer Interval	60	1200

## HFM Web Tuning Parameters

This section covers other HFM Web UI tuning parameters.

## Tuning Parameter and Locations

Parameter	Default Value	Suggested Value
		
<p><b>HFM ADF Web App Java Heap Size</b> (Xms and Xmx)</p> <p>The amount of heap available to each managed instance of the HFM ADF Web Application.</p> <p><b>Windows</b></p> <p>HFM ADF Web is installed as a Windows service. The heap parameters can be tuned by editing the JVMOptionXX options in the following location:</p> <p>HKLM\SOFTWARE\Hyperion Solutions\HFMWeb (N) \HyS9FinancialMangementWeb</p> <p>Where <b>HFMWeb(N)</b> is HFMWeb0, HFMWeb1, etc. depending on how many HFM web servers are deployed and which server instance you are updating.</p> <p>Note that on a compact deployment, it is not possible to tune HFM ADF Web separately as there will be a single EPMSysyem(N) Web application for all EPM products. The location of the Windows registry containing configuration options is</p> <p>HKLM\SOFTWARE\Hyperion Solutions\EPMServer (N) \HyS9EPMServer.</p>	128m	8192m
<p>Xms sets the initial heap size and should be set to the same size as Xmx.</p> <p><b>Windows</b></p> <p>Locate JVMOptionXX for -Xms</p> <p>Data Type is REG_SZ</p> <p>Data value: -Xms&lt;size&gt;</p>		

Parameter	Default Value	Suggested Value
Xmx sets the maximum size of the heap. On 32bit systems, set no higher than 1.8 GB, and on 64bit it is recommended not to set higher than 75% of the <i>available</i> physical memory. <b>Windows</b> Locate JVMOptionXX for -Xmx Data Type is REG_SZ Data value: -Xmx<size>	4096m	8192m

## Tuning Financial Management Applications

In general, running more than one Oracle Hyperion Financial Management application at a time on a single application server impacts the performance of all applications. Do not attempt to run more than three or four applications at a time, even if other applications are idle, because idle applications require database connections and CPU time to run.

### Commonly Tuned Financial Management Settings

A complete list of configuration settings can be found in Appendix A of this guide. See [Configuration Settings](#).

#### MaxNumDataRecordsInRAM

In general, MaxNumDataRecordsInRAM is the most significant setting, as it decides how many records to maintain in RAM. The number of records in memory is checked when either of the two conditions NumMinutesBeforeCheckingLRU or NumCubesLoadedBeforeCheckingLRU is reached. See [Available Configuration Settings](#).

Range: No hard limit on range.

Default value: 30,000,000



**Note:**

When total number of records in RAM goes above this value, the FreeLRU is called to release records from memory to free up some memory for the Oracle Hyperion Financial Management server. The informational message "FreeLRUCachesIfMoreRAMIsNeeded released data cubes" is recorded in the log.

#### MinDataCacheSizeInMB

By setting this value at a higher number, you can reduce the number of DataCache growth attempts and thus reduce memory fragmentation. Typically DataCache is grown on a per-need basis, and grows 25 MB maximum at a time.

Range: No hard limit on range.

Default value: 2,000

## MaxDataCacheSizeInMB

This setting controls the maximum amount of memory that the Oracle Hyperion Financial Management application server allocates to store the cell values and cell status. If more memory is required by the system, then the cell value and cell status are paged out to disk based on the LRU logic. The workaround is to increase the cache size.

Range: The value should be greater than or equal to 500.

Default value: 4,500

### Note:

If performance of a consolidation operation decreases because of paging, you can increase MaxDataCacheSizeInMB to minimize paging. This value should be more than the total memory usage allowed by "MaxNumDataRecordsInRAM", so that the system does not page out the cells unnecessarily to disk. A too low MaxDataCacheSizeInMB setting means that you will run out of memory to store data records and begin paging, which reduces system performance.

## MaxNumCubesInRAM

This setting controls the number of cubes that are held in memory at any given time. The number of cubes in memory is checked when either of the two conditions NumMinutesBeforeCheckingLRU or NumCubesLoadedBeforeCheckingLRU is reached. Lowering this setting can alleviate high memory usage for sparse applications (applications with many entities, but relatively few records per entity). This setting affects all data operations including consolidations and data retrievals.

Range: 100-500,000

Default value: 60,000

## IcmSystemReportTranslationMode

When running an Intercompany Matching System Report that requires currency translations, translated values are written to the database by default. This can lead to longer running reports and a significant increase in database size. This setting allows for control over how translated values are handled.

Range: 0,1,2

Default value: 0

Valid Data options:

- 0 - Default behavior; translated values are written to the database.
- 1 - On-the-fly calculations use Sub Translate in rules, but no translated values are written to the database. Note: Parent Entities in CN status will also be translated.
- 2 - On-the-fly calculations only execute default translation (no Sub Translate rule). No changes are written to the database. Note: Parent Entities in CN status will not be translated.

## NumConsolidationThreads

This setting controls the multi-threading of consolidations per application server.

Range: No hard limit on range. Value must be a positive number.

Default value: 8

### Note:

Lowering the value limits the system's utilization of system resources, resulting in slower consolidation performance. Raising the value results in higher CPU utilization and may affect other components' performance.

### Tip:

Tests were conducted to evaluate the impact of increasing NumConsolidationThreads from 4 to 8. Results below show that one month consolidation times became faster.

Transactions	92 users NumConsolidationThreads = 4 Average Response Time (seconds)	92 users NumConsolidationThreads = 8 Average Response Time (seconds)
01_Run_Consolidation_A_1105	10.11	3.22
02_Run_Consolidation_A_0005	16.15	9.47
03_Run_Consolidation_A_2205	7.75	3.19
04_Run_Consolidation_A_3305	18.67	9.17
05_Run_Consolidation_B_1105	8.21	3.14
06_Run_Consolidation_B_0005	13.26	9.27
07_Run_Consolidation_B_2205	7.69	6.20
08_Run_Consolidation_B_3305	18.29	9.41
09_Run_Consolidation_C_0005	30.59	22.08

### Note:

CPU utilization on Financial Management application servers during test execution was acceptable with capacity available to accommodate more intense workload.

### Note:

Before increasing this value, ensure all settings are the same on all application servers; you should test to see how many current consolidations will run on a given server before total consolidation time is actually worse when running concurrent consolidations versus consolidations waiting in queue.

## MaxNumConcurrentConsolidations

This controls the number of concurrent consolidations allowed per application server. Any consolidations executed above the value are queued as Scheduled Consolidations.

Range: 1 - no limit

Default value: 8

Scenario: If you have three Oracle Hyperion Financial Management application servers, each server could run 8 maximum number of concurrent consolidations, but the default value of NumConsolidationsAllowed limits you to running only 8 concurrent consolidations total on the three servers against one application. Example: Users submit six consolidations on ServerA, then users submit two more consolidations on ServerB; these 8 consolidations will all run. At the same time, if users submit the next consolidation on ServerC, it will not run until one of the previous 8 finish (in the Running Tasks page, it will have a status of Scheduled Start).

 **Note:**

Before increasing this value, ensure all registry settings are the same on all application servers; you should test to see how many current consolidations will run on a given server before total consolidation time is actually worse when running concurrent consolidations versus consolidations waiting in queue.

## NumThreadsToUseWhenUpdatingCalcStatusSystemWasChanged

During the metadata load, when the entity hierarchy changes (members are moved, added or deleted), then in-use members may become inconsistent with their children or parents. Oracle Hyperion Financial Management must verify calculation status for each scenario and year combination. This setting enables multiple threads instead of a single thread to verify and update calculation status.

Range: 1 - no limit

Default value: 16

## Financial Management Memory Settings for Larger Applications

The following table contains suggested values for parameters depending on available physical memory. This is done with the assumption that Oracle Hyperion Financial Management is the only memory-intensive process running on the machine and running only one Financial Management application.

 **Note:**

If multiple applications will be active, then divide the total physical memory installed on the server by the number of applications to arrive at the Available Physical Memory for each application.

Available Physical Memory	MaxNumDataRecordsInRAM	MaxDataCacheSizeinMB
8 GB	10,000,000	1,500
16 GB	30,000,000	4,500
32 GB	60,000,000	9,000
64 GB	100,000,000	15,000
128 GB	200,000,000	30,000

## Application-Specific Settings

Some settings that previously were only environment-level settings have been expanded to an application level. Typically, tuning Oracle Hyperion Financial Management at the application level is beneficial when several applications must run on the same Financial Management application server, but only one application is heavily used. In this case, the heavily used application is tuned to enable it to use the bulk of the server's memory and the remaining applications simply use the default values, limiting the amount of memory the application can utilize. To use the Financial Management application-specific or server-specific settings, use the Override procedure described in the Configuration Settings chapter. See [Overriding Values](#).

### Note:

The override order of precedence is as follows:

1. If an application-specific setting does not exist and an installation registry setting does, the installation registry setting is used.
2. If an application-specific setting does not exist, the setting defined in the Server key is used.
3. If no application-specific or server setting is defined, the default value is used. For settings and default values, see [Configuration Settings](#).

These settings can be overridden with application-specific settings:

- AllowOverlappingConsolidationOverride
- AutoClearDeadTasks
- AutoClearEAFlatfileTasksAfterSeconds
- EnableRulesLogging
- EnableRunningTasksMaskUserNames
- DSStartupOption
- MaxDataCacheSizeInMB
- MaxNumConcurrentConsolidations
- MaxNumCubesInRAM
- MaxNumDataRecordsInRAM
- MaxNumRetriesOfBaseLevelCalculation
- MinDataCacheSizeInMB
- NumConsolidationThreads

- NumCubesLoadedBeforeCheckingLRU
- NumDataLoadsAllowed
- NumEAExportsAllowed
- NumEAThreads
- NumMinutesBeforeCheckingLRU
- NumThreadsToUseWhenUpdatingCalcStatusSystemWasChanged

## Tuning Financial Management Application Servers

When tuning Oracle Hyperion Financial Management application servers, you should start with baseline tests to measure key user activities with representative user concurrency. When using multiple Financial Management clusters, often to separate reporting and Oracle Smart View for Office user activity from consolidation activity, it is likely to see different tuning changes affect each server differently based on the user task being measured. For example, an Financial Management application server primarily used for reporting will see no benefit by increasing NumConsolidationThreads, while a server running many consolidations should see improvements in consolidation times. Likewise, an application server primarily used for reporting would likely see better response times for repeated reports when MaxNumDataRecordsInRAM is set high enough to keep all records in memory, while a server running many consolidations is unlikely to see consolidation times improve. Another point to consider when deciding what role an application server will have is CPU speed, CPU core count, RAM amount and RAM speed. An application server primarily dedicated to running consolidations, running one Financial Management application that has intensive rules, will typically see the best performance with faster CPU clock speeds, with at least 8 cores, rather than using more CPUs/cores, but slower clock speed.

## Financial Management Tuning Example

This example describes tuning an HFM application server with 8 CPUs, 64 GB of RAM and hosting three HFM applications with only one of the three being used heavily; call this application CORP. The Oracle Hyperion Financial Management application server will be used for both consolidation and reporting activity.

When you install Financial Management, standard configuration settings are provided by default. You can view these default settings and modify them in the Settings module.

In this example, we need to keep the total of the MaxNumDataRecordsinRAM value to 60,000,000, and the total MaxDataCacheSizeinMB value to 9,000. Since two of the three applications are not heavily used, we will allow those two application to use default values, and tune the CORP application to take full advantage of the server resources.

MaxNumDataRecordsinRAM default value is 30,000,000, with two applications using defaults - that means CORP can have MaxNumDataRecordsinRAM set to 40,000,000.

MaxDataCacheSizeinMB default value is 4,500, with two applications using defaults - that means CORP can have MaxDataCacheSizeinMB set to 6,000.

The other value we would consider tuning specifically for CORP in this example would be:

MinDataCacheSizeInMB – setting to half the value of MaxDataCacheSizeinMB (3,000).



Since we are tuning at the application level for CORP and allowing the other two applications to use default values, we create overrides for CORP to use values different than the defaults. These would be the updates to the settings:

MaxNumDataRecordsinRAM (40,000,000)

MaxDataCacheSizeinMB (6,000)

MinDataCacheSizeinMB (4,000)

To update these configuration settings, select **Navigate**, then **Administer**, then **Consolidation Administration**, then **System**, and select **Settings**

For information on all configuration settings, see [Available Configuration Settings](#).

for information on settings for large applications, see [Financial Management Memory Settings for Larger Applications](#).

## Application Database Maintenance

The following best practices are recommended when using Oracle / SQL Server databases with Oracle Hyperion Financial Management:

For Financial Management tables <appname>\_DATA\_AUDIT, <appname>\_TASK\_AUDIT, and HFM\_ERRORLOG, it is recommended to implement the following best practices:

Quarterly - Business to review the Audit logs, archive and delete.

Half-Yearly - Archive System Messages and truncate table.

### Tip:

Set up alerts so that action can be taken if these tables grow beyond the recommended number of records (> 500,000). Note that large audit tables can have severe impact on Financial Management performance.

## Basic Design Considerations

- If Data Audit feature is not part of business requirements, then it is recommended to turn off data auditing. Degradation in performance has been observed for applications with a Data Audit table more than 10GB.

### Tip:

To turn off auditing of members, change the EnableDataAudit attribute to N for all members in your metadata file.

- Rules must always be tested prior to loading in a production environment to avoid any pitfalls of poorly designed rules (for example, data explosion from rules). Efficient rules are critical for acceptable system performance.

- Oracle Hyperion Financial Management does all processing through subcubes while they are stored in RAM, so the larger the subcube, the larger the impact on performance. Try to minimize subcube size as much as possible, for example, no subcube should exceed the 200,000 base record limits in order to ensure optimal system performance.
- Loading to or calculating zeros in a Financial Management application is not recommended. Zeros are stored as data, which increases the database size and can affect performance. It is recommended that only numeric information, such as 1000, be stored in Financial Management. See the Data File section in the *Oracle Hyperion Financial Management User's Guide*.
- If Financial Management is not properly shut down, temporary files may remain upon reboot. To ensure optimal performance, it is recommended that you delete all \*.db.\* file names from the Financial Management Server Working folder before launching Financial Management.
- For attaching multiple documents to a data grid or process unit, Oracle recommends that you attach no more than three documents to a cell. Each document should be smaller than 100K to limit the performance effect on the database.

 **Tip:**

You can set a size limit for document attachments and a maximum number of document attachments by user when you create an application. You can set the limits in the AppSettings attribute for the application metadata.

- Use the Consolidate All option only under appropriate circumstances. If this option is used, the system does not skip entities with NoData, which can have a significant impact on consolidation performance.

 **Tip:**

The Consolidate (Impacted Consolidation) is the most efficient option, because only entities that require logic or consolidation are updated by the system. The Consolidate All with Data option is useful for updating system status from OK SC to OK after metadata changes. For additional information, see the Consolidation Options section of the *Oracle Hyperion Financial Management User's Guide*.

## Tuning Oracle 11g Databases for Financial Management

### Introduction

Oracle Hyperion Financial Management requires a relational database to store application data. Each Financial Management application contains a set of tables, indexes, stored procedures, and other objects. Because the number and sizes of these objects vary according to the user's data set, application design, and reporting requirements, it is difficult to specify a concrete set of rules for setting up the database. This section addresses the two most common issues that arise during deployment to Oracle databases:

- Oracle DB running out of memory to support the required number of database connections
- Poor performance during reporting and consolidation

Both of these issues can be traced to improper Oracle parameter settings and configurations. Creating a System Global Area (SGA) that is too large limits the amount of free physical memory to support user connections and activities. Conversely, creating an SGA that is too small causes additional disk access, and slows down performance.

This section guides you through the process of monitoring a typical database and determining the proper initialization settings to maximize performance. You should repeat this process periodically to keep up with changes to your data set, workload, and application design.

Oracle 11g has made the process of monitoring and tuning the database much easier than previous versions. We strongly recommend using Oracle Enterprise Manager (both Data Control and Grid Control) to monitor, diagnose, and tune database performance. To obtain accurate instance statistics, Oracle recommends that you enable Oracle database Automatic Maintenance Tasks.

It is very important that tuning not be done immediately after database startup. At that point, the buffer caches are empty and no statistics have been collected. Always test and tune your database after a period of activity on Financial Management applications.

## Common Performance Issues

The most common cause of poor performance is stale or missing table statistics. Oracle Hyperion Financial Management applications create new tables or can clear existing tables and reload or replace entire data sets on the fly. By default, Oracle 10g and 11g run nightly jobs to check for tables that need statistics to be updated. This may not be sufficient in some cases. The following activities are common end user activities where the DBA should be informed that schema statistics should be updated:

- New HFM application created and loaded
- Data in an existing Scenario is cleared and reloaded
- New Scenario/Year is opened and loaded with data. This includes scenarios populated by Financial Management rules.

After a Scenario/Year has been loaded with the first month's data, data loads to subsequent periods do not affect table statistics.

## Tuning Guidelines for Oracle 11g Databases

### Oracle Initialization Parameters

Many initialization parameters can be fine-tuned to improve database performance. This section focuses on the parameters that are known to greatly influence Oracle Hyperion Financial Management performance with an Oracle database.

### CURSOR\_SHARING

`CURSOR_SHARING` determines what type of SQL statements can share the same cursors. If this parameter is set to `EXACT`, only statements with identical text can share the same cursor. If this parameter is set to `FORCE`, statements that differ in some literals but are otherwise identical can share and reuse SQL cursors, unless the literals affect the meaning of the statement. Tests show that setting this parameter to a value of `FORCE` may improve consolidation and reporting performance significantly. This is because with this parameter set to `FORCE`, the Oracle database spends less time parsing SQL statements, and requires less memory.

Suggested setting for all releases prior to 11.1.2.2.300: `FORCE`

Suggested setting for releases 11.1.2.2.300 and later: EXACT

## MEMORY\_TARGET

MEMORY\_TARGET and MEMORY\_MAX\_TARGET are two new parameters in Oracle database release 11g. These two parameters determine the use of Automatic Memory Management for an Oracle database. Oracle strongly recommends the use of Automatic Memory Management to manage the memory on your system. By setting these two parameters to non-zero values, Oracle enables Automatic Memory Management, and tunes to the target memory size, redistributing memory as needed between system global area (SGA) and the instance program global area (PGA). As a result, the following parameters are automatically sized:

SGA\_TARGET

SGA\_MAX\_SIZE

DB\_CACHE\_SIZE

SHARED\_POOL\_SIZE

LARGE\_POOL\_SIZE

JAVA\_POOL\_SIZE

STREAMS\_POOL\_SIZE

PGA\_AGGREGATE\_TARGET

However, LOG\_BUFFER is not affected by Automatic Memory Management, and still needs to be manually sized. Tuning LOG\_BUFFER is discussed later in this section.

Since MEMORY\_TARGET specifies the total memory size of SGA and PGA, it should be set to a relatively high value to achieve better performance. Oracle Hyperion Financial Management consolidation and reporting are memory-intensive tasks, and require abundant memory. Considering that 32-bit operating systems have limits on available address space (typically 2 GB or 3 GB), Oracle recommends this parameter to be set to at least 1.2 GB. Generally, higher values are associated with better Oracle database performance, so we would like to set this parameter as high as possible without running out of virtual address space.

Suggested setting: minimum 1.2GB, generally higher than 1.2GB (depending on the environment).

When determining the amount of memory to allocate to the Oracle 11g instance, review this section: [How to Calculate the Number of Processes for Oracle Database Release 11g](#).

## MEMORY\_MAX\_TARGET

MEMORY\_MAX\_TARGET specifies the maximum value to which a DBA can set the MEMORY\_TARGET parameter. It serves as an upper limit so MEMORY\_TARGET cannot be set too high accidentally. It also reserves memory for the Oracle database instance, in case you want to increase MEMORY\_TARGET at runtime without a restart. Therefore, MEMORY\_MAX\_TARGET should be no lower than MEMORY\_TARGET.

Suggested setting: No lower than MEMORY\_TARGET

## SGA\_TARGET

SGA\_TARGET specifies the total size of all SGA components. If Automatic Memory Management is enabled, and SGA\_TARGET is set to a non-zero value, then this value serves as the minimum size of SGA.

Suggested setting: 0 if Automatic Memory Management is enabled; otherwise refer to [How to Determine Memory Settings for Oracle Database Release 11g](#).

## SGA\_MAX\_SIZE

SGA\_MAX\_SIZE specifies the maximum size of the SGA for the lifetime of the instance. This parameter sets the upper limit for SGA\_TARGET. If Automatic Memory Management is enabled, the Oracle database cannot increase the total size of SGA components beyond SGA\_MAX\_SIZE.

Suggested setting: Use the default setting if Automatic Memory Management is enabled; otherwise refer to [How to Determine Memory Settings for Oracle Database Release 11g](#).

## PGA\_AGGREGATE\_TARGET

PGA\_AGGREGATE\_TARGET specifies the total PGA memory available to all server processes attached to the instance. If Automatic Memory Management is enabled, and PGA\_AGGREGATE\_TARGET is set to a non-zero value, then this value serves as the minimum size of PGA.

Suggested setting: 0 if Automatic Memory Management is enabled; otherwise refer to [How to Determine Memory Settings for Oracle Database Release 11g](#).

## LOG\_BUFFER

LOG\_BUFFER specifies the amount of memory (in bytes) that Oracle uses when buffering redo entries to a redo log file. Redo log entries contain a record of the changes that have been made to the database block buffers. Oracle Hyperion Financial Management is a high-update transactional system, and the database constantly uses log buffer. Properly sizing the log buffer can improve the database performance. In general, larger values for LOG\_BUFFER reduce redo log file I/O, particularly if transactions are long or numerous. If the buffer is too small, the system waits until the buffer is clear before adding new updates, so it is important to set this buffer correctly to improve database performance.

Suggested setting: Start with 8 MB. Refer to [How to Determine Memory Settings for Oracle Database Release 11g](#).

## OPTIMIZER\_MODE

OPTIMIZER\_MODE establishes the default behavior for choosing an optimization approach for the instance. You can set the optimizer mode to FIRST\_ROWS for optimal online application response, or to ALL\_ROWS to minimize total execution time for batch operations. Because Oracle Hyperion Financial Management deals only with the total returned data set, minimizing the total execution time is more appropriate.

Suggested setting: ALL\_ROWS

## OPTIMIZER\_INDEX\_COST\_ADJ

OPTIMIZER\_INDEX\_COST\_ADJ allows you to tune optimizer behavior for access path selection to be more or less index friendly; that is, to make the optimizer more or less prone to selecting an index access path over a full table scan. The default for this parameter is 100 percent, at which the optimizer evaluates index access paths at the regular cost. But Oracle Hyperion Financial Management transactions generally favor index access paths more than full table scan paths, so Oracle recommends a lower value for this parameter.

Suggested setting: 50

## PROCESSES

PROCESSES specifies the maximum number of operating system user processes that can simultaneously connect to an Oracle database. Because Oracle Hyperion Financial Management only works with dedicated servers, each connection requires a process. Each Financial Management application requires a minimum of 200 (400 when in multi-server environments) database connections in addition to the number specified by the Financial Management database connection pool setting.

Suggested setting: Refer to [How to Calculate the Number of Processes for Oracle Database Release 11g](#).

## SESSIONS

This parameter specifies the maximum number of sessions that can be created on the database system. Because every login requires a session, this parameter effectively determines the maximum number of concurrent users on the Oracle database. The default value is  $1.1 * PROCESSES + 5$ . Oracle does not recommend setting this parameter below its default value.

## TRANSACTIONS

TRANSACTIONS specifies the maximum number of concurrent transactions. Because some transactions may be recursive, this parameter should be greater than SESSIONS, and in turn, PROCESSES, to allow for recursive transactions. The default value is  $1.1 * SESSIONS$ . Oracle does not recommend setting this parameter below its default value.

## OPEN\_CURSORS

OPEN\_CURSORS specifies the maximum number of open cursors (handles to private SQL areas) that a session can have at once. It is important to set the value of OPEN\_CURSORS high enough to prevent your application from running out of open cursors. Assuming that a session does not open the number of cursors specified by OPEN\_CURSORS, there is no added overhead to setting this value higher than actually needed.

Suggested setting: 5000

## SESSION\_CACHED\_CURSORS

SESSION\_CACHED\_CURSORS specifies the number of session cursors to cache. Repeated parse calls of the same SQL statement cause the session cursor for that statement to be moved into the session cursor cache. Subsequent parse calls will find the cursor in the cache, and do not need to reopen the cursor. Performance of Oracle Hyperion Financial Management

applications will benefit from this cache because Financial Management connections are also cached.

Suggested setting: 50

## TRACE\_ENABLED

TRACE\_ENABLED controls tracing of the execution history, or code path, of Oracle database. Enabling this option by setting the parameter to TRUE adds additional overhead on the database and is not recommended for a normal Financial Management application environment.

Suggested setting: FALSE

## STATISTICS\_LEVEL

STATISTICS\_LEVEL specifies the level of statistics collection for database and operating system. The Oracle database collects these statistics for a variety of purposes, including making self-management decisions. The default setting of TYPICAL ensures collection of all major statistics required for database self-management functionality, and provides the best overall performance.

Suggested setting: TYPICAL

## TIMED\_STATISTICS

TIMED\_STATISTICS specifies whether statistics related to time are collected. Starting with Oracle database release 11.1.0.7.0, the value of the TIMED\_STATISTICS parameter cannot be set to FALSE if the value of STATISTICS\_LEVEL is set to TYPICAL or ALL.

Suggested setting: TRUE

## TIMED\_OS\_STATISTICS

TIMED\_OS\_STATISTICS specifies (in seconds) the interval at which Oracle collects operating system statistics when a request is made from the client to the server or when a request completes. Enabling this option by setting the parameter to a number greater than 0 severely degrades the performance of applications.

Suggested setting: 0

## How to Determine Memory Settings for Oracle Database Release 11g

This section outlines how to monitor and view Oracle system-related statistics, and tune Oracle database memory parameters. There are many ways to determine optimal memory settings, but the preferred way is to use memory advisors, including Memory Advisor, SGA Advisor, Shared Pool Advisor, Buffer Cache Advisor, and PGA Advisor. You must have an Oracle login with DBA privileges to use these advisors and to perform the following tasks. Note that most of the queries below have equivalent graphical interfaces through Oracle Enterprise Manager.

### Total Memory Sizing (MEMORY\_TARGET)

MEMORY\_TARGET specifies the Oracle system-wide usable memory, including both SGA and PGA. Prior to Oracle database release 11g, SGA and PGA had to be tuned separately.

If a database is upgraded from Oracle 10g to 11g, MEMORY\_TARGET can be determined by simply adding SGA\_TARGET and PGA\_AGGREGATE\_TARGET from the Oracle 10g database.

If a database is upgraded from Oracle 9i to 11g, MEMORY\_TARGET can be determined by adding PGA\_AGGREGATE\_TARGET and all SGA components, including DB\_CACHE\_SIZE, SHARED\_POOL\_SIZE, LARGE\_POOL\_SIZE, JAVA\_POOL\_SIZE, etc.

If a database has not been upgraded from an earlier version, and has no history references, Oracle recommends this parameter to be initially set to 1 to 3 GB, depending on system resources and system limits. After the database has been in use for some time, this parameter can be tuned as follows. (This tuning also applies to the above two upgrade scenarios).

```
SQL> select * from v$memory_target_advice order by memory_size;
```

```
MEMORY_SIZE MEMORY_SIZE_FACTOR ESTD_DB_TIME ESTD_DB_TIME_FACTOR VERSION
```

```
180 .5 458 1.344 0
```

```
270 .75 367 1.0761 0
```

```
360 1 341 1 0
```

```
450 1.25 335 .9817 0
```

```
540 1.5 335 .9817 0
```

```
630 1.75 335 .9817 0
```

```
720 2 335 .9817 0
```

The row with the MEMORY\_SIZE\_FACTOR of 1 shows the current size of memory, as set by the MEMORY\_TARGET initialization parameter, and the amount of DB time required to complete the current workload. In previous and subsequent rows, the results show a number of alternative MEMORY\_TARGET sizes. For each alternative size, the database shows the size factor (the multiple of the current size), and the estimated DB time to complete the current workload if the MEMORY\_TARGET parameter were changed to the alternative size. Notice that for a total memory size smaller than the current MEMORY\_TARGET size (360 in this example), estimated DB time (ESTD\_DB\_TIME) increases. Notice also that in this example, there is nothing to be gained by increasing total memory size beyond 450MB, because the ESTD\_DB\_TIME value is not decreasing. Therefore, in this example, the suggested MEMORY\_TARGET size is 450 MB.



## SGA Sizing (SGA\_TARGET)

Normally, SGA is tuned automatically by the Oracle database if Automatic Memory Management is enabled. But a DBA can still monitor the size of SGA and find out if it is at the optimal size.

```
SQL> select * from v$sga_target_advice order by sga_size;
```

```
SGA_SIZE SGA_SIZE_FACTOR ESTD_DB_TIME ESTD_DB_TIME_FACTOR ESTD_PHYSICAL_READS
```

```
290 .5 448176 1.6578 1636103
```

```
435 .75 339336 1.2552 1636103
```

```
580 1 201866 1 513881
```

```
725 1.25 201866 1 513881
```

```
870 1.5 201866 1 513881
```

```
1015 1.75 201866 1 513881
```

```
1160 2 201866 1 513881
```

Based on a similar analysis in the Total Memory Sizing section, the current setting of SGA\_TARGET is already optimal.

## PGA Sizing (PGA\_AGGREGATE\_TARGET)

Similar to SGA, PGA is also automatically tuned by the Oracle database if Automatic Memory Management is enabled. The following query can be used to monitor if PGA size is properly sized. The result is shown similar to the query results of v\$memory\_target\_size and v\$sga\_target\_size.

```
SQL> select * from v$pga_target_advice order by pga_target_for_estimate;
```

## LOG\_BUFFER Sizing

In the system view `v$sysstat`, the value for redo buffer allocation retries reflects the number of times a user process waits for space in the redo log buffer. This value should be near zero for a properly sized database. For example:

```
select name, value

from v$sysstat

where name = 'redo buffer allocation retries'

NAME VALUE

redo buffer allocation retries 1021967
```

If the log buffer has no space for updates, the database must wait and retry. In this example, the database has retried a total of 1,021,967 times. To improve performance, increase the `LOG_BUFFER` parameter value. This value is expressed in bytes and must be a multiple of the log block size value, which is the operating system block size. For the Oracle Hyperion Financial Management application, set `LOG_BUFFER` to a minimum of 8 MB, and then use the above query to monitor the performance and increase as necessary. If `LOG_BUFFER` needs to be increased, Oracle recommends growing it by 50% at a time.

## How to Calculate the Number of Processes for Oracle Database Release 11g

The number of user processes that can simultaneously connect to the Oracle database is limited by the `PROCESSES` initialization parameter of the Oracle database instance. By default, each Oracle Hyperion Financial Management application process requires a minimum of 200 database connections for a single server environment. When more than one Financial Management server has been configured for an environment, an additional pool is required for Cluster Controller connections to ensure application integrity. The Cluster Controller Pool is equal to the Application pool, therefore 400 connections per application process must be accounted for by the DBA.

### Note:

The total number of servers in a cluster, and the total number of applications, affect the number of required database connections.

This example illustrates how to calculate the number of processes that connect to the Oracle database. Suppose a single Financial Management application server has two applications. The default Financial Management database connection pool setting is 200. The minimum number of Oracle database connections required for just Financial Management is 400. For additional safety, multiply this number by a factor of 1.1 to allow for ancillary connections and

general usage of the database. Considering that the Oracle database also has some background processes, add 20 to the number to arrive at the value of PROCESSES. So, in this case, PROCESSES should be set to 460.

For single HFM applications server:

$PROCESSES = (\text{Financial Management connection pool setting}) * (\text{Number of Financial Management applications}) * 1.1 + 20.$

Suppose a cluster of two Financial Management application servers has two applications on each server. The Financial Management database connection pool setting is 200 and the Cluster Controller pool would also be 200. The minimum number of Oracle database connections required for just Financial Management is now 1,600:  $(200+200)*2*2=1600.$

For Multi-Server environment:

$PROCESSES = (\text{Financial Management connection pool setting} + \text{Financial Management Cluster Controller connection pool setting}) * (\text{Number of Financial Management applications}) * (\text{Number of Financial Management servers}) * 1.1 + 20.$

## Other Considerations

### Shared Server versus Dedicated Server

Oracle Hyperion Financial Management requires all connections to the Oracle database to be served by dedicated server processes. In other words, Financial Management does not work with shared server processes. Dedicated server processes consume more CPU and memory resources but achieve better performance. To use a dedicated server, the net service name value should include the SERVER=DEDICATED clause in the connect descriptor. This is an example of a net service configured for dedicated server processes:

```
HFMDB = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP) (HOST = hfm.oracle.com)
(PORT = 1521)) (CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME = HFMDB1) ) )
```

### Online Redo Log Files Size

The size of the redo log files can influence performance, because the behaviors of the database writer and archiver processes depend on the redo log sizes. Generally, larger redo log files provide better performance. Undersized log files increase checkpoint activity and reduce performance. However, small log files and frequent checkpoints reduce recovery time. So, if daily operational efficiency is more important than minimizing recovery time, then set online redo log files to a relatively large value. Hundreds of MB is a normal size for Oracle Hyperion Financial Management databases. But the preferred way to determine the size of redo log files is to enable FAST\_START\_MTTR\_TARGET, and run a typical database workload for a while. Then run the following query to obtain the optimal size of redo logs.

```
SQL> select optimal_logfile_size from v$instance_recovery;
```

For more details about how to tune MTTR target and the size of online redo files, please refer to the *Oracle Database Performance Tuning Guide*.

### Tablespaces and Segments Fragmentations

Over time, updates and deletes on objects within a tablespace can create pockets of empty space that individually are not large enough to be reused for new data. This type of empty

space is referred to as fragmented free space. Objects with fragmented free space can result in much wasted space, and can impact database performance. Oracle Hyperion Financial Management consolidation performs extensive updates, inserts, and deletes, so it is very important to monitor the fragmentation of tablespaces and defragment them regularly. The preferred way to defragment and reclaim this space is to perform an online segment shrink. For more information about how to use online segments, please refer to the Oracle Database Administrator's Guide or consult Oracle database support services

## Index Fragmentation

Oracle Hyperion Financial Management applications usually create hundreds or even thousands of indexes. As application data changes over time, indexes could become fragmented. Regular monitoring and defragmentation of these indexes can improve performance. However, index rebuild is a time-consuming and resource-intensive operation. Oracle does not recommend any index rebuild while applications are in operation. Oracle Enterprise Manager provides user-friendly interfaces to monitor the statistics of indexes. For more details about how to monitor and defragment indexes through Enterprise Manager, refer to Oracle database documentation.

## Disable the feature DEFERRED\_SEGMENT\_CREATION

Oracle introduced the DEFERRED\_SEGMENT\_CREATION feature in Release 11.2. The default setting is On in all installations. The feature ensures that a TABLE create statement does not actually create a table. The table is only created after a row of data is inserted.

## Disable the feature DEFERRED\_SEGMENT\_CREATION

Oracle introduced the DEFERRED\_SEGMENT\_CREATION feature in Release 11.2. The default settings is On in all installations. With this feature, a TABLE create statement does not actually create a table. The table is only created after a row of data is inserted. This feature can cause issues when exporting and importing Oracle Hyperion Financial Management schemas, as some tables may not be created during the import. It is recommended that this feature be disabled; tables must then created automatically. To disable this feature, log in to your instance using SYSTEM or SYS and issue the command:

```
alter system set deferred_segment_creation=false;
```

Any table created after issuing this statement is created automatically. If you already have an instance with empty tables and want to export the application, you can alter each table individually to force creation and allow the table to be used by the EXP (export) command.

To determine if a schema has empty tables, run either of the two following commands:

```
select segment_name, segment_type, extents from dba_segments where extents <
1 and segment_type='TABLE' and owner='<hfm db schema>'
```

```
select table_name from all_tables where owner='<hfm db schema>' and
table_name not in (select segment_name from dba_segments where owner='<hfm db
schema>' and segment_type='TABLE' and extents>0)
```

Issue the following command for each empty table:

```
alter table <table_name> allocate extent
```

## Regular Maintenance and Tuning Plans

The preceding sections outline the typical process to correctly size the Oracle memory parameters. Performance tuning, by its nature, is iterative. Removing one performance block might not lead to immediate performance improvement because another block might be revealed. Therefore, this process should be repeated until the performance is acceptable. Because Oracle Hyperion Financial Management application data changes constantly from period to period, regular database maintenance and tuning plans will help users proactively monitor and tune Oracle database performance, and prevent potential performance issues in the future. For more information and additional tuning options, consult Oracle database support.

## Frequently Asked Questions

### Which Operating Systems are supported?

See the Oracle Enterprise Performance Management System Supported Platform Matrix: [Oracle EPM Supported Platform Matrix](#).

### Must the relational database be 64-bit?

The database can be either 32-bit or 64-bit, as long as it is a supported DBMS type and version.

Considerations must be made for third-party and extending software. By default, the Oracle Hyperion Financial Management installation only installs 64-bit software on a 64-bit operating system. This means that only the 64-bit client components are installed on the Financial Management application server. When 32-bit connectivity is required, components may not function unless the 32-bit client software is installed on the 64-bit application server. For more information, refer to the *Oracle Enterprise Performance Management System Installation and Configuration Guide*.

### What are the memory limitations of 64-bit Financial Management?

In practical terms, 64-bit Financial Management is limited by physical memory, rather than virtual memory. It can take advantage of all available physical memory after the proper memory parameter adjustments are made.

### Are there any memory settings that must be tuned for Financial Management?

Financial Management default memory settings are appropriate for a small to medium size application in a 32-bit environment. To take full advantage of available memory, Oracle recommends the following settings for a monthly application. The relevant settings are `MaxNumDataRecordsInRAM` and `MaxDataCacheSizeinMB`, which should be changed. The following table contains suggested values for these parameters depending on available memory. The table assumes that Financial Management is the only memory-intensive process running on the machine and running only one Financial Management application. If multiple applications are active, then divide the Total physical Memory installed on the server by the number of Financial Management applications, to determine the Available Physical Memory for each application.

Available Physical Memory	MaxNumDataRecordsinRAM	MaxDataCacheSizeinMB
4	4,000,000	500
8	10,000,000	1500

Available Physical Memory	MaxNumDataRecordsInRAM	MaxDataCacheSizeinMB
16	30,000,000	4500
32	60,000,000	9000

Example: On a server with 24 GB of RAM with 2 active monthly Financial Management applications, the MaxNumDataRecordsInRAM value should be 22,500,000 and the MaxDataCacheSizeinMB value should be 3375.

On average, the amount of memory used for one data record in a monthly application is 112 bytes, a weekly application uses 472 bytes per data records and a daily application uses 3,296 bytes per data record.

For a Weekly application, divide the MaxNumDataRecordsInRAM by 4, without changing the value in the last column for MaxDataCacheSizeinMB.

For Daily applications, divide the MaxNumDataRecordsInRAM by 30, without changing the value in the last column for MaxDataCacheSizeinMB.