Chapter 1. About Financial Management

- Financial Management Features ........................................... 15
- Architecture ........................................................................ 16
- Performance Management Architect ...................................... 16
- Workspace ........................................................................... 16

Chapter 2. Managing Applications

- Application Administration .................................................... 23
- Creating Application Profiles ................................................ 24
  - Entering Languages .......................................................... 25
  - Defining Calendars ......................................................... 26
  - Editing Frequencies ........................................................ 27
  - Editing Periods ............................................................... 28
- Saving Application Profiles .................................................... 32
- Creating Applications .......................................................... 32
- Opening Applications ......................................................... 35
- Closing Applications .......................................................... 36
- Deleting Applications .......................................................... 36
- Copying Financial Management Applications .......................... 37
- Working with System Messages ............................................ 39
  - Retrieving System Messages ............................................ 40
<table>
<thead>
<tr>
<th>Chapter 3. Managing Application Security</th>
<th>55</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Security Considerations</td>
<td>56</td>
</tr>
<tr>
<td>Launching the User Management Console from Financial Management</td>
<td>56</td>
</tr>
<tr>
<td>Assigning Users and Groups to Financial Management Applications</td>
<td>57</td>
</tr>
<tr>
<td>Setting Up Security Classes for Financial Management Applications</td>
<td>58</td>
</tr>
<tr>
<td>Assigning User Access to Security Classes</td>
<td>59</td>
</tr>
<tr>
<td>Setting Up E-mail Alerting</td>
<td>60</td>
</tr>
<tr>
<td>Running Security Reports for Financial Management Applications</td>
<td>62</td>
</tr>
<tr>
<td>Loading Application Security</td>
<td>63</td>
</tr>
<tr>
<td>Clearing and Loading Security Information</td>
<td>65</td>
</tr>
<tr>
<td>Before Clearing Security Information</td>
<td>65</td>
</tr>
<tr>
<td>After Clearing Security Information</td>
<td>66</td>
</tr>
<tr>
<td>Extracting Application Security</td>
<td>68</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 4. Managing Metadata</th>
<th>71</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metadata Attributes</td>
<td>72</td>
</tr>
<tr>
<td>Defining Accounts</td>
<td>75</td>
</tr>
<tr>
<td>Account Type Behavior</td>
<td>78</td>
</tr>
<tr>
<td>Defining Dynamic Accounts</td>
<td>80</td>
</tr>
<tr>
<td>Defining Custom Members</td>
<td>80</td>
</tr>
<tr>
<td>Defining Entity Members</td>
<td>82</td>
</tr>
<tr>
<td>Defining Scenario Members</td>
<td>83</td>
</tr>
<tr>
<td>Defining Application Settings</td>
<td>85</td>
</tr>
<tr>
<td>Organization by Period</td>
<td>88</td>
</tr>
<tr>
<td>Defining Consolidation Methods</td>
<td>89</td>
</tr>
</tbody>
</table>
# Contents

<table>
<thead>
<tr>
<th>Chapter 8. Using Extended Analytics</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Extended Analytics Registry Settings</td>
<td>188</td>
</tr>
<tr>
<td>Star Schema</td>
<td>188</td>
</tr>
<tr>
<td>Star Schema Formats</td>
<td>189</td>
</tr>
<tr>
<td>Creating and Exporting Data to a Star Schema</td>
<td>191</td>
</tr>
<tr>
<td>Updating a Star Schema</td>
<td>192</td>
</tr>
<tr>
<td>Deleting a Star Schema</td>
<td>193</td>
</tr>
<tr>
<td>Creating a Star Schema Template</td>
<td>193</td>
</tr>
<tr>
<td>Deleting a Star Schema Template</td>
<td>194</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 9. Defining Reports</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Defining Reports</td>
<td>195</td>
</tr>
<tr>
<td>Data Explorer Report Example</td>
<td>196</td>
</tr>
<tr>
<td>Creating Journal Reports</td>
<td>197</td>
</tr>
<tr>
<td>Journal Report Example</td>
<td>197</td>
</tr>
<tr>
<td>Creating Intercompany Partner Matching Reports</td>
<td>198</td>
</tr>
<tr>
<td>Selecting Member Lists for Intercompany Matching Reports</td>
<td>198</td>
</tr>
<tr>
<td>Selecting Accounts for Intercompany Matching Reports</td>
<td>198</td>
</tr>
<tr>
<td>Specifying Decimal Places in Intercompany Matching Reports</td>
<td>199</td>
</tr>
<tr>
<td>Selecting Style Sheets for Intercompany Matching Reports</td>
<td>199</td>
</tr>
<tr>
<td>Specifying Currencies in Intercompany Matching Reports</td>
<td>199</td>
</tr>
<tr>
<td>Suppression Options for Intercompany Matching Reports</td>
<td>199</td>
</tr>
<tr>
<td>Defining Intercompany Partner Matching Report Scripts</td>
<td>200</td>
</tr>
<tr>
<td>Creating Intercompany Transaction Reports</td>
<td>207</td>
</tr>
<tr>
<td>IC Transaction Report Syntax</td>
<td>207</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 10. Managing Rules</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule Types</td>
<td>211</td>
</tr>
<tr>
<td>Rule Considerations</td>
<td>212</td>
</tr>
<tr>
<td>Calculation Commands</td>
<td>213</td>
</tr>
<tr>
<td>Current Dimension Members</td>
<td>214</td>
</tr>
<tr>
<td>Account Expressions</td>
<td>217</td>
</tr>
<tr>
<td>Functions Automatically Clear Data</td>
<td>218</td>
</tr>
<tr>
<td>Error Messages</td>
<td>219</td>
</tr>
<tr>
<td>Rule Execution During Consolidation</td>
<td>220</td>
</tr>
</tbody>
</table>
IsFirst ............................................................ 291
IsICP ............................................................. 292
IsLast ............................................................. 293
IsTransCur ......................................................... 293
IsTransCurAdj ...................................................... 294
IsValidDest ......................................................... 294
IsZero ............................................................ 295
List .............................................................. 296
Member ........................................................... 297
MemberFromID ..................................................... 298
Method ........................................................... 299
NoInput ........................................................... 299
NoRound .......................................................... 300
NumBase .......................................................... 301
Number ........................................................... 302
NumChild .......................................................... 303
NumDescendant ..................................................... 304
OpenDataUnit ...................................................... 306
PCon ............................................................. 307
PlugAcct .......................................................... 307
POwn ............................................................ 308
PVAForBalance ..................................................... 309
PVAForFlow ........................................................ 310
RateForBalance ..................................................... 310
RateForFlow ........................................................ 311
ReviewStatus ....................................................... 311
ReviewStatusUsingPhaseID ........................................ 312
Round ............................................................ 313
Scale ............................................................. 314
SecurityAsPartner ................................................... 315
SecurityClass ....................................................... 316
SetData ........................................................... 317
SetDataWithPOV ................................................... 318
SubmissionGroup .................................................... 319
SupportsTran ....................................................... 319
SwitchSign ......................................................... 319
SwitchType ........................................................ 321
Trans ............................................................. 322
TransPeriodic ....................................................... 322
Chapter 14. Managing Process Management Submission Phases ........................................ 351
Defining Submission Phases by Scenario and Period ...................................................... 351
Submission Group Assignments by Scenario and Period ............................................... 353
Assigning Submission Groups to Phases ...................................................................... 355
Viewing Unassigned Submission Groups ..................................................................... 356

Glossary .................................................................................................................. 357

Index ....................................................................................................................... 363
Financial Management Features

Oracle’s Hyperion® Financial Management – System 9 supports these features:

- A single, unified view of enterprise financial information consolidates key performance and operating metrics from global sources in a scalable, Web-based application.
- “Virtual close” features trim days and weeks off your close cycle using Web-based intercompany reconciliations and a consistent set of data and business measures.
- Powerful multidimensional analysis identifies and reports 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 all 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 as well as for 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 specific 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.
Architecture

Financial Management is designed to operate as a multitier system.

- The client tier contains the user interface and the ability to communicate with the application tier. You can display data and metadata, enter data, and maintain metadata in this tier.
- On the Web server tier, you access the Web-enabled portions of Financial Management.
- The middle tier contains the domain intelligence and connections to the relational database.
- The data tier contains the relational database and all Financial Management data and metadata.

Performance Management Architect

Oracle’s Enterprise Performance Management Architect is an optional component of Financial Management installation and configuration. You use it to create and work with applications and dimensions, and synchronize data.

For help on tasks performed in Performance Management Architect, see the Hyperion Enterprise Performance Management Architect Administrator’s Guide.

Workspace

Financial Management is available within Oracle’s Hyperion® Workspace. For information on tasks performed in Oracle’s Hyperion® Workspace, such as Workspace preferences or features in the Navigate, Favorites, Manage, or Tools menu, see the Hyperion Workspace User’s Guide, Administrator’s Guide and Web 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. Financial Management supplies eight system-defined dimensions and enables you to populate up to four custom dimensions that you can apply to accounts.

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 the child of a parent member. All members below a parent are referred to as descendants. The bottom-level hierarchy members are called base-level members.

The following sections describe the system-defined dimensions. For information on setting dimension attributes, see the Hyperion Enterprise Performance Management Architect Administrator’s Guide if you are using Performance Management Architect, or the Hyperion Financial Management — System 9 Administrator’s Guide if you are using Classic Application Administration.
**Scenario Dimension**

The Scenario dimension represents a set of data, such as Budget, Actual, 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. Financial Management supports years, months, and weeks for the period dimension. It does not support days for the 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 are dependent, base, or parent entities. Dependent entities are owned by other entities in the organization. Base entities are at the bottom of the organization structure and do not own other entities. 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 different 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 its parent entity.

**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. 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; for example, 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**

Four custom dimensions are available for analysis of detailed data. You can use custom dimensions to store 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.

**User-Defined Elements**

Many elements in Financial Management are user-defined. For example, when you create a journal, you give the journal a label and a 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>
<thead>
<tr>
<th>Table 1</th>
<th>Requirements for User-Defined Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element</strong></td>
<td><strong>Min. length</strong></td>
</tr>
<tr>
<td>Application Profile</td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>1</td>
</tr>
</tbody>
</table>
| Period label | 1 | 20 | *Must contain only alphanumeric characters.*  
*Cannot start with a number.*  
*Cannot contain spaces, symbols, or diacritical marks such as umlauts.* |
| View label | 1 | 10 | *Must contain only alphanumeric characters.*  
*Cannot start with a number.*  
*Cannot contain spaces, symbols, or diacritical marks such as umlauts.* |
| View description | 0 | 40 | Cannot contain an ampersand ( & ). |
| Period description | 0 | 40 | Cannot contain an ampersand ( & ). |
| Create Application | | | |
| Application label | 1 | 10 | *Must contain only alphanumeric characters.*  
*Cannot start with a number.*  
*Cannot contain spaces, symbols, or diacritical marks such as umlauts.*  
**Note:** Application labels are not case-sensitive. For example, App1 and APP1 are considered the same application label. |
| Application description | 1 | 40 | Cannot contain an ampersand ( & ). |
| Metadata Manager | | | |
| Member label | 1 | 20 | Must be unique. The label can contain up to 20 characters including spaces, but cannot start with a space.  
**Note:** If you are using an Oracle database, member labels cannot contain spaces.  
You cannot use the following characters in the member name:  
* period ( . )  
* plus sign ( + )
<table>
<thead>
<tr>
<th>Element</th>
<th>Min. length</th>
<th>Max. length</th>
<th>Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td></td>
<td></td>
<td>● minus sign ( - )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>● asterisk ( * )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>● forward slash ( / )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>● number sign ( # )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>● comma ( , )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>● semicolon ( ; )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>● at symbol ( @ )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>● double quotation mark ( &quot; )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>● curly brackets ( { } )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>● ampersand ( &amp; )</td>
</tr>
<tr>
<td>Note:</td>
<td></td>
<td></td>
<td>You cannot use ALL as the name of an entity.</td>
</tr>
<tr>
<td>Member description</td>
<td>0</td>
<td>40</td>
<td>Cannot contain an ampersand ( &amp; ).</td>
</tr>
<tr>
<td>Alias label</td>
<td>0</td>
<td>20</td>
<td>Cannot contain an ampersand ( &amp; ).</td>
</tr>
<tr>
<td>Security</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security class</td>
<td>1</td>
<td>80</td>
<td>None</td>
</tr>
<tr>
<td>Journals</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Journal label</td>
<td>1</td>
<td>20</td>
<td>Cannot include the following characters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>● period ( . )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>● plus sign ( + )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>● minus sign ( - )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>● asterisk ( * )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>● forward slash ( / )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>● number sign ( # )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>● comma ( , )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>● semicolon ( ; )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>● at symbol ( @ )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>● double quotation mark ( &quot; )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>● curly brackets ( { } )</td>
</tr>
<tr>
<td>Note:</td>
<td></td>
<td></td>
<td>If you are using an Oracle database, journal labels cannot contain spaces.</td>
</tr>
<tr>
<td>Journal description</td>
<td>0</td>
<td>255</td>
<td>None</td>
</tr>
<tr>
<td>Journal group</td>
<td>0</td>
<td>30</td>
<td>None</td>
</tr>
<tr>
<td>Journal line item description</td>
<td>0</td>
<td>50</td>
<td>None</td>
</tr>
<tr>
<td>Load/Extract</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Element</td>
<td>Min. length</td>
<td>Max. length</td>
<td>Restrictions</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-------------</td>
<td>-------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Delimiter character</td>
<td>1</td>
<td>1</td>
<td>Must be one of the following characters and cannot be used in the file or in the file name:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- comma ( , )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- tilde ( ~ )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- at sign ( @ )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- number sign ( # )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- dollar sign ( $ )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- percent sign ( % )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- ampersand (&amp;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- carat ( ^ )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- line (</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- colon ( : )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- semicolon ( ; )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- question mark ( ? )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- back slash ( \ )</td>
</tr>
<tr>
<td>Note: The ampersand ( &amp; ) is not a valid delimiter for metadata .app files. 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.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Data Explorer

<table>
<thead>
<tr>
<th>Element</th>
<th>Min. length</th>
<th>Max. length</th>
<th>Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell description</td>
<td>1</td>
<td>1900</td>
<td>None</td>
</tr>
<tr>
<td>Line item detail</td>
<td>1</td>
<td>80</td>
<td>None</td>
</tr>
<tr>
<td>Annotation</td>
<td>0</td>
<td>255</td>
<td>None</td>
</tr>
<tr>
<td>Decimal character</td>
<td>1</td>
<td>1</td>
<td>The following characters are invalid decimal characters for Data Explorer:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- back slash ( \ )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- forward slash ( / )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- plus sign ( + )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- minus sign ( - )</td>
</tr>
</tbody>
</table>

Documents

<table>
<thead>
<tr>
<th>Element</th>
<th>Min. length</th>
<th>Max. length</th>
<th>Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document names (including folder and report names)</td>
<td>1</td>
<td>16</td>
<td>The following characters are invalid characters for document names:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- plus sign ( + )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- question mark ( ? )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- asterisk ( * )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- back slash ( \ )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- forward slash ( / )</td>
</tr>
<tr>
<td>Element</td>
<td>Min. length</td>
<td>Max. length</td>
<td>Restrictions</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>number sign (#)</td>
<td></td>
<td></td>
<td>• number sign (#)</td>
</tr>
<tr>
<td>comma (,)</td>
<td></td>
<td></td>
<td>• comma (,)</td>
</tr>
<tr>
<td>semicolon (;)</td>
<td></td>
<td></td>
<td>• semicolon (;)</td>
</tr>
<tr>
<td>colon (:)</td>
<td></td>
<td></td>
<td>• colon (:)</td>
</tr>
<tr>
<td>at sign (@)</td>
<td></td>
<td></td>
<td>• at sign (@)</td>
</tr>
<tr>
<td>double quotation mark (&quot; )</td>
<td></td>
<td></td>
<td>• double quotation mark (&quot; )</td>
</tr>
<tr>
<td>curly brackets ({ })</td>
<td></td>
<td></td>
<td>• curly brackets ({ })</td>
</tr>
<tr>
<td>line (</td>
<td>)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>less than sign (&lt;)</td>
<td></td>
<td></td>
<td>• less than sign (&lt;)</td>
</tr>
<tr>
<td>greater than sign (&gt; )</td>
<td></td>
<td></td>
<td>• greater than sign (&gt;)</td>
</tr>
<tr>
<td>period (.) at the end of a document name</td>
<td></td>
<td></td>
<td>• period (.) at the end of a document name</td>
</tr>
</tbody>
</table>

**Note:** Document names also cannot contain trailing or leading white space.
Managing Applications

In This Chapter

Application Administration ................................................................. 24
Creating Application Profiles .......................................................... 25
Creating Applications ......................................................................... 32
Opening Applications ......................................................................... 35
Closing Applications ........................................................................... 36
Deleting Applications ......................................................................... 36
Copying Financial Management Applications ....................................... 37
Working with System Messages .......................................................... 39
Managing System Users ...................................................................... 41
Managing Servers and Applications .................................................... 43
Auditing Tasks .................................................................................... 45
Auditing Data ..................................................................................... 48
Monitoring Running Tasks ................................................................. 50
Scanning For and Deleting Invalid Records ......................................... 51
Changing Settings for Multiple Server Tasks ....................................... 52
Maintaining Server Logon Information ............................................... 53

Caution!

The information in this chapter about application administration is provided for use with Classic Financial Management applications only. For information on application administration using Performance Management Architect, see Hyperion Enterprise Performance Management Architect Administrator’s Guide.

The following roles are required for application administration. For information on assigning roles, see the Hyperion Security Administration Guide.

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension Editor</td>
<td>Creates and manages import profiles for dimension creation, as well as creating and managing dimensions manually within the Performance Management Architect user interface or the Classic Application Administration option.</td>
</tr>
<tr>
<td>Role</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>Required to access Classic Application Administration options for Financial Management and Planning using Web navigation.</td>
</tr>
<tr>
<td></td>
<td>Creates and deploys Performance Management Architect applications. Users with this role can create applications, but can change only the dimensions to which they have access permissions.</td>
</tr>
<tr>
<td></td>
<td>Required in addition to the Dimension Editor role for Financial Management and Planning users to be able to navigate to their product’s Classic Application Administration options.</td>
</tr>
<tr>
<td></td>
<td>When a user with Application Creator role deploys an application from Performance Management Architect, that user automatically becomes the application administrator and provisioning manager for that application. The Application Creator can create all applications.</td>
</tr>
<tr>
<td></td>
<td>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 Financial Management Configuration Utility.</td>
</tr>
</tbody>
</table>

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.

Applications run on application servers. You can set up clusters of application servers to balance the load on multiple servers. For instructions, see the *Hyperion Financial Management — System 9 Installation Guide*.

**Application Administration**

You define an application in five steps:

1. Create an application profile, which contains calendar, language, and frequency definitions for the application. See “Creating Application Profiles” on page 25.

2. Create an application shell, in which you define the application server where the application resides, an application label and description, an application profile, and a location for a log file. See “Creating Applications” on page 32.

3. Define security for the application including which users have access to the application and what type of access each user has. See Chapter 3, “Managing Application Security.”

4. Define metadata for the application, including accounts, entities, scenarios, and custom dimensions, as well as application settings, consolidation methods, and currencies. See Chapter 4, “Managing Metadata.”

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

---

24 Managing Applications
You can have only one application open at a time, however, you can have several instances of Financial Management open with different applications on each instance.

Creating Application Profiles

An application profile contains language, calendar, frequency, and period information for an application.

You can use a profile for more than one application. For each application that you create, you must specify an application profile.

Note:
You create application profiles in the Financial Management Desktop in Windows.

For more information on creating application profiles, see these topics:

- Entering Languages
- Defining Calendars
- Editing Frequencies
- Editing Periods
- Saving Application Profiles

➤ To create an application profile:
1. From the Financial Management Windows desktop, select Define Application Profile.
2. Select one of the following options:
   - Create a New Application Profile
   - Select an Existing Application Profile and click to locate the file.

Note:
By default, application profile files use the PER file extension.
3. Click Next.

Entering Languages

You can define up to 10 languages for labels that are used throughout an application. You can use different languages to create descriptions for items in your application.

➤ To set up languages:
1. In the grid, enter a language, and press the Tab key to move down to the next line.
Defining Calendars

When you select the type of calendar and the time periods for the application profile, default frequencies are created for the application profile. For example, if you select standard calendar and include half-years, quarters, and months as the time periods, the system creates the following frequencies: yearly, half-yearly, quarterly, and monthly.

You can also select a custom calendar or manual calendar. If you choose to define a custom calendar, you must define the number of periods and the period label prefix.

A flat list is created and you can modify the hierarchy later. If you choose to define a manual calendar, the frequencies and periods are empty. You can then enter the frequencies and periods that you need.

➤ To set up a calendar:

1 Select a type of calendar:
   - Standard Calendar
   - Custom Calendar
   - Manually Defined Calendar

2 Do one of the following:
   - If you selected Standard Calendar:
     ☐ Select the time periods to include (half-years, quarters, trimesters, or months).
     ☐ If you included months in the calendar, from the Start Month drop-down list, select the first month in the calendar.
If you selected Custom Calendar:

- In the Number of Base Periods text box, enter the number of periods in the year.
- In the Period Label Prefix text box, enter a prefix for the periods to include.
  
  The label can contain a maximum of 10 characters and can include spaces.

**Tip:**

If you enter 10 for the number of periods and NewPeriod as the label prefix, the following periods are added to the hierarchy: NewPeriod1, NewPeriod2, NewPeriod3, NewPeriod4, NewPeriod5, and so on.

- If you selected Manually Defined Calendar, continue with the next step.

3 In the Start Year text box, enter the start year for the calendar.

4 In the Number of Years text box, enter the total number of years to include.

5 Click Next.

### Editing Frequencies

Frequencies and their corresponding views 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.

When editing frequencies, you cannot change the label of the YTD frequency, which is contained in the first column of Frequency 1 row. However, you can enter a description of the YTD frequency 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 Manually Defined Calendar as the time period for the calendar, the Frequencies grid is empty, and you must enter the necessary frequency 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 set up frequencies:

1 From the Language drop-down list, select a language.

2 Enter or edit the values in the View column and enter a View description.
You can enter up to six frequencies.

**Note:**

The View label can contain a maximum of 20 characters. The View description can contain a maximum of 40 characters. Labels cannot include these characters:

\[ . + - \ast / \# \{ \} ; , @ " \]

**Tip:**

Because you cannot modify frequencies after an application is created, make sure to include a view 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 achieve a 13-month year.

For instructions on adding sibling and child periods to the hierarchy, see “Adding Sibling and Child Periods” on page 29.

When you select a period, you can view the period frequency and change the period label or description.

**Note:**

You cannot edit period labels based on language. In other words, 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 enter and edit periods:

1. **From the period hierarchy, select a period, and, in the Description text box, enter or edit the period description.**

   The period description can contain a maximum of 40 characters. Note that a space is counted as a character. You cannot edit the description for the Year period.

   **Tip:**

   Make sure to include a description for each period when you are creating an application profile. If you do not include a description and choose to add one later, you will need to modify the application profile and re-create all applications that use the profile.

2. **In the Label text box, enter or edit the period label.**
The period label can contain a maximum of 20 characters. Note that a space is counted as a character. You cannot edit the label for the Year period. Changes that you make to the period label are reflected in all languages defined in the application profile.

**Tip:**

You can use and to expand or collapse the levels in the period hierarchy.

3 From the Language drop-down list, select a language.

You can have a unique period description for each language. You cannot have a unique period label for each language.

4 Click Next.

### Adding Sibling and Child Periods

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

- To add a single child or sibling period:
  1. Highlight the period to which to add a child or sibling period.
  2. Right-click the period, and do one of the following:
     - Select Insert Sibling.
     - Select Insert Child.

  **Tip:**

  You can also use the insert child toolbar button and the insert sibling toolbar button.

  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. Right-click the period, and select Insert Multiple.

3. Select one of the following options:
   - to add multiple sibling periods to the selected period
   - to add multiple child periods to the selected period

4. Enter the number of periods to add to the hierarchy.
5. Enter a label prefix for the new periods.

Tip:
If you enter 10 for the number of periods and NewPeriod as the label prefix, the following periods are added to the hierarchy: NewPeriod1, NewPeriod2, NewPeriod3, NewPeriod4, NewPeriod5, and so on.

Moving Periods
You can use the mouse to drag and drop periods. You can move periods to sibling or child positions in the hierarchy.

To move periods:

1. Highlight a period to move.
2. Click and drag the period to the new location in the hierarchy.
3. When you release the mouse button, select one of the following options:
   - Move Here (Sibling) to move the period and make it a sibling of the highlighted period
   - Move Here (Child) to move the period and make it a child of the highlighted period
Note:
You can also copy the period as a sibling or child of the selected period. When you release the mouse button, select the Copy Here (Sibling) or Copy Here (Child) option.

Copying Periods
You can copy single and multiple periods in the period hierarchy. You can copy a selected period and all child periods of the selected period. You can paste a copied period as a sibling or child of the selected period.

➤ To copy periods:
1 Highlight a period to copy.
2 Right-click the highlighted period and select one of the following options:
   ● Copy Period to copy only the highlighted period
   ● Copy All to copy the highlighted period and all child periods of the highlighted period
3 Right-click the period to which to paste the copied period and select one of the following options:
   ● Paste as Sibling to paste the copied period as a sibling of the selected period
   ● Paste as Child to paste the copied period as a child of the selected period
4 In the Label text box, make changes to the period label.
   The label can contain a maximum of 20 characters. Note that a space is counted as a character.
5 In the Description text box, enter a description for the period.
   The description can contain a maximum of 40 characters. Note that a space is counted as a character.

Deleting Periods
When you delete periods from the hierarchy, all children of the period are also deleted.

Note:
You cannot delete the Year period.

➤ To delete a period, take one of the following actions:
   ● Right-click the period and select Delete.
   ● Highlight the period and click }
Saving Application Profiles

By default, application profile files use the PER file extension. If you are creating an application profile, you must complete all steps in the Design Application Profile module.

➤ To save an application profile:

1. In the File Name text box, enter a name for the file, or click ![browse](image) to browse for a profile to overwrite.

   Note:
   By default, application profile files use the PER extension.

2. Click Finish.

Creating Applications

Caution!
This information is provided for use with Classic Financial Management applications only. For information on application administration using Performance Management Architect, see Hyperion Enterprise Performance Management Architect Administrator's Guide.

You create an application by specifying the following information:

- The application server cluster on which to run the new application (The cluster name must be a valid server name.)
- The application label
- The application description
- The application profile, which contains calendar, language, and frequency definitions for the application
- The local storage folder, which stores some application information on your workstation
- The user management project to which to add the application.

See the Hyperion Security Administration Guide.

- The Web server URL for security administration

   Note:
   After you create an application, you must set up security for it. See Chapter 3, “Managing Application Security.”
Windows Procedure

To create an application:

1. From the desktop navigation frame, select **Create Application**.

2. From the **Server** drop-down list, select the application server cluster on which to run the application.

   If the server cluster is not listed, you may need to register the server cluster. See the *Hyperion Financial Management — System 9 Installation Guide*.

3. In the **Application Label** text box, enter a name for the application.

   The application label can have a maximum of 10 alphanumeric characters but cannot start with a number or contain spaces or special characters. 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.

4. In the **Application Description** text box, enter a description for the application.

   The application description can have a maximum of 40 characters, and can include spaces, but cannot contain an ampersand (&) character. HFM, HSV, and HSX are reserved names and cannot be used for application descriptions.

5. In the **Application Profile** text box, enter the path and file name of the application profile, or click ![browse](image) to browse for the application profile to use.

   By default, application profiles use the PER file extension.

6. In the **Local Storage Folder** text box, enter the path and the folder name where downloaded application files will be stored on the workstation, or click ![browse](image) to browse to the local storage folder.

7. From the **User Management Project** drop-down list, select the Shared Services project to which to add the application.

   **Note:**

   Each application must belong to a Shared Services project. See the *Hyperion Security Administration Guide*.

8. In the **Financial Management Web Server URL for Security Administration** text box, enter the URL.

9. Click **Create**.

10. Click **Close**.
Web Procedure

➤ To create an application:

1 Select Navigate > Administer > Classic Application Administration > Consolidation Administration.

2 Select Create Application.

3 From the Server drop-down 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 Hyperion Financial Management — System 9 Installation Guide.

4 In the Application Name text box enter a name for the new application.

The application label can have a maximum of 10 alphanumeric characters but cannot start with a number or contain spaces or special characters. 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.

Caution!

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

5 In the Application Description text box, enter a description for the application.

The application description can have a maximum of 40 characters, and can include spaces, but cannot contain an ampersand (&) character. HFM, HSV, and HSX are reserved names and cannot be used for application descriptions.

6 Click Browse next to the Application Profile text box, and locate the application profile to use.

By default, application profiles use the PER file extension.
7 From the User Management Project drop-down list, select the Shared Services project to which to add the application.

Note:
Each application must belong to a User Management project. See the Hyperion Security Administration Guide.

8 In the Financial Management Web Server URL for Security Administration text box, enter the URL.

9 Click Create.

Opening Applications

You can open one application for each Financial Management session that you are running. However, you can have several Financial Management sessions open, with different applications open on each session.

You must specify a local working folder for each application. You specify the working folder for an application when the application is created. If the application was created on a computer other than the one that you are using, you need to specify a local working folder the first time you open the application.

Note:
To access an application, you must be assigned as a user of the application. See “Assigning Users and Groups to Financial Management Applications” on page 57.

Windows Procedure

➤ To open an application:

1 From the desktop navigation frame, select Open Application.

2 Click Connect.

When you connect to an application server or cluster, the system authenticates your user name and password. If you enter an invalid username-password combination, you cannot access or create applications.

3 From the Cluster list, select the application server or cluster on which the application is running.

4 Highlight the application to open, and click Open Application.

Web Procedure

➤ To open an application:

1 Select Navigate > Applications > Consolidation > AppName, where AppName is the name of the application.
2 From the Select Cluster list, select the application server on which the application is running.

When you connect to an application server or cluster, the system authenticates your username and password. If you enter an invalid username-password combination, you cannot access or create applications.

If the server cluster is not listed, you may need to register it. See the Hyperion Financial Management — System 9 Installation Guide.

3 Select the application to open.

Closing Applications

When you close an application, the system prompts you to save the changes that you made.

Note:

If you close an application and then want to re-open it, wait approximately 10 seconds before reopening the application to provide time for the Financial Management services to shut down properly.

Windows Procedure

➤ To close an application:
   1 From the desktop navigation frame, select Close Application.
   2 Click Yes to close the application.

Web Procedure

➤ To close an application:
   1 Select File > Close Application.
   2 Click Yes to close the application.

Deleting Applications

Caution!

This information is provided for use with Classic Financial Management applications only. For information on application administration using Performance Management Architect, see Hyperion Enterprise Performance Management Architect Administrator's Guide.

Before deleting an application, be sure that no other user is currently using the application.
Note:
To delete an application, the user must be the Application Administrator for that application. Additionally, for the Web, the user also needs the global roles of Dimension Editor and Application Creator.

Windows Procedure

➤ To delete an application:
1. From the desktop navigation frame, select Delete Application.
2. From the Cluster drop-down list, select the application server cluster on which the application is running.
   If the server cluster is not listed, you may need to register the server cluster. See the Hyperion Financial Management — System 9 Installation Guide.
3. Click Connect.
4. Select an application, and click Delete Application.
5. Click Yes to delete the application.
6. Click Close.

Web Procedure

➤ To delete an application:
1. Select Navigate > Administer > Classic Application Administration > Consolidation Administration.
2. Select Delete Application.
   If you have the application open, you must close it before you can delete it.
3. Select the application server on which the application is running.
   If the server cluster is not listed, you may need to register it. See the Hyperion Financial Management — System 9 Installation Guide.
4. Select the application to delete.
5. Click Yes to delete the application.

Copying Financial Management Applications

Caution!
This information is provided for use with Classic Financial Management applications only. For information on application administration using Performance Management Architect, see Hyperion Enterprise Performance Management Architect Administrator’s Guide.
Financial Management provides a utility to copy and move an application from one application server to the same server or another application server. An application can be copied across databases. For example, an application created in a SQL database can be copied to an Oracle database.

The copy application utility, HFMCopyApplication.exe, is installed to the following default location:

C:\Hyperion\FinancialManagement\Server

**Note:**

Before copying an application, make sure that all users are disconnected from the application that is being copied and that all processing is stopped. Otherwise, the copied application may contain incorrect data.

After you copy an application, you must register the new application with Shared Services and then set up a user with Application Administrator role for the new application. For information on registering applications, see “Configuring and Setting Up Financial Management” in the *Hyperion Financial Management — System 9 Installation Guide*. For information on assigning user roles, see “Managing and Provisioning Users and Groups” in the *Hyperion Security Administration Guide*.

➢ To copy an application:

1. In Windows Explorer, navigate to the Server directory of your Financial Management installation, and double-click HFMCopyApplication.exe.
2. At the Welcome screen, click Next.
3. Click ▶ to locate the source database link file (UDL) for the application you want to copy and click Next.
4. From the list of applications, select the application to copy and click Next.
5. Click ▶ to locate the destination database link file (UDL) and click Next.
6. From the drop-down list, select the name of the application to which you want to copy or enter a new application name and click Next.
7. On the Options panel, select any or all of the following actions and click Next.
   - Copy Application Data to copy data from the source application to the destination application.
   - Copy Audit Data to copy audit data from the source application to the destination application.
   - Copy Cluster Settings to copy cluster information from the source application to the destination application.
   - Overwrite Existing Application (if it exists) to overwrite an application of the same name as the destination application.
     - Drop All application tables prior to copy
o Only drop tables that are being copied

- Optional: Click Advanced Options for additional options.
  a. Modify these values as required:
    - Use Client-side Cursor or Use Server-Side Cursor
    - Use SQL Binding
    - Use default thread count or Use custom thread count
    - Log SQL errors
    - Number of task retries
  b. Click OK to close the Advanced Options box.

8 Confirm the settings and click Next.

9 When processing completes, click OK.

10 To view a task detail either in Completed tasks or Failed tasks (if any), double-click the task, or select the task and click the question mark (?) icon.

11 Click Close.

12 Click Finish to exit, or View Log File to view the log file and see any errors that occurred during the copy operation.

Note:
An improperly sized Oracle instance can cause the utility to fail to copy all table rows. If the error log displays an Oracle issue, or table rows are missing, you may need to increase the size of the REDO logs.

13 After you copy an application, register the new application with Shared Services and set up a user with administrative rights for the new application.

### Working with System Messages

You can retrieve, view, print, and delete system messages for Financial Management. You can retrieve system messages by date, server name, and application name. System messages remain in the log until you delete them. The system messages module is available only in the Web user interface.

Note:
Security is not implemented on the system messages log; therefore, the log can be accessed and cleared by any user.

For more information on system messages, see these procedures:

- “Retrieving System Messages” on page 40
- “Viewing and Printing System Messages” on page 40
Retrieving System Messages

To retrieve system messages:
1. Select Administration > System Messages.
2. Optional: You can filter messages using these criteria:
   - Start date
   - End date
   - Server name
   - Application name

Note:
You can leave the server name and application name blank to view system messages for all servers and applications.
3. Click View.

Viewing and Printing System Messages

To view and print system messages:
1. In the System Messages log, select the check box next to a message to view details.

Note:
To view detail for a system message, click the relevant cell in the System Message Summary column.
2. Click Details.
3. Click Print to print the messages displayed.
Deleting System Messages

To delete system messages, do one of the following:

- To delete individual messages, select the check box next to each message to delete, and click Delete.
- To delete all system messages, click Delete All.

System Message Detail Strings

Some methods 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 the following example:

```
Error Reference Number: {219EB33B-BF50-11D6-A43E-0000863DCCF1}
Num: 0x800415c6; Type: 1; DTime: 9/3/02 12:20:10 PM; Svr: GSZAB01; File: CHsxServerImpl.cpp; Line: 1842; Ver: 3.0.0.196;
```

The following table describes the fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Num</td>
<td>The error number in hexadecimal form</td>
</tr>
<tr>
<td>Type</td>
<td>For internal use only</td>
</tr>
<tr>
<td>DTime</td>
<td>The TimeStamp of the error.</td>
</tr>
<tr>
<td>Svr</td>
<td>The machine name of the computer on which the error occurred</td>
</tr>
<tr>
<td>File</td>
<td>The name of the source code file to which the error applies</td>
</tr>
<tr>
<td>Line</td>
<td>The line number in the source code file to which the error applies</td>
</tr>
<tr>
<td>Ver</td>
<td>The version number of the DLL to which the error applies</td>
</tr>
</tbody>
</table>

Managing System Users

The Users on System feature enables you to view the users on the system and log off users of an application or server. You can view what module the user is working in and what activities are being performed by the user. See “Viewing Users” on page 42 and “Logging Off Users” on page 42.

To view and log off users by application or server, you need not be logged on to the application. However, to log off individual users, you must be logged on to the application.
Only members of the Administrator group that you specify when you configure the application server can manage users on the system. See the *Hyperion Financial Management — System 9 Installation Guide.*

**Note:**

Users on System is only available in the Financial Management Web user interface.

**Viewing Users**

You can view logged-on users for all applications and all servers, logged-on users of a specific application on a specific server, logged-on users of a specific application on all servers, and logged-on users of all applications running on a specific server.

The following information is available for each logged-on user:

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

➤ To view logged-on users:

1. Select Administration > Users on System.
2. Specify the users to view by completing one of the following actions:
   - To view all logged-on users for all applications and all servers, leave Server Selection and Application Selection blank.
   - To view all logged-on users for all applications on a server, enter the name of the server in the Server Selection text box and leave the Application Selection text box blank.
   - To view all logged-on users of an application, enter the name of the application in the Application Selection text box and leave the Server Selection text box blank.
   - To view all logged-on users of an application on a server, enter the name of the server in the Server Selection text box and the name of the application in the Application Selection text box.
3. Click View Logged-In Users.

**Logging Off Users**

To perform a system-wide process, such as a backup and restore, you can log users off an application or an application server. When you log off a user, processes that the user is executing finish before the user is logged off. You can log off users by application or by 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” on page 44.

➤ To log users off an application or server:

1 Select Administration > Users on System.

2 Set options:
   ● Enter the name of the server from which to log off users.
   ● Enter the name of the application from which to log off users.

   Note:
   If you leave these text boxes blank, the system assumes ALL for the text box. For example, if you enter a server name but leave the application text box blank, all users of all applications on the specified server are logged off.

3 Click Logout Users.

➤ To log individual users off an application or server:

1 Open the application from which to log off users.

2 Select Administration > Users on System.

3 Specify the users to view by completing one of the following actions:
   ● To view all logged-on users for all applications and all servers, leave Server Selection and Application Selection blank.
   ● To view all logged-on users of all application on a server, enter the name of the server in the Server Selection text box and leave the Application Selection text box blank.
   ● To view all logged-on users of an application, enter the name of the application in the Application Selection text box and leave the Server Selection text box blank.
   ● To view all logged-on users of an application on a server, enter the name of the server in the Server Selection text box and the name of the application in the Application Selection text box.

4 Select View Logged-In Users.

5 In the Users on System list, click ➡ in the Logout column to log an individual user off the application.

6 Click OK to confirm.

Managing Servers and Applications

You can use the Manage Servers and Applications feature to disable user connections to an application or server and to log off users.

See “Disabling and Enabling Connections” on page 44 and “Logging Off Users” on page 42.
Only members of the Administrator group that you specify when you configure the application server can manage servers and applications. See the Hyperion Financial Management — System 9 Installation Guide.

**Note:**
The Manage Servers and Applications feature is available only in the Financial Management Web user interface.

**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 off user feature. For example, you can disable logging on to an application, log off users logged on to the application, load metadata, and then enable connections to the application. For more information on the log off feature, see “Logging Off Users” on page 42.

**Note:**
Administrators can perform all administrative tasks, such as loading metadata and deleting applications, when connections are disabled.

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

1. Select Administration > Manage Servers and Applications.
2. You can perform these actions:
   - To disable or enable user connections for all applications and all servers, leave Server Selection and Application Selection blank.
     
     **Note:**
     This also disables connections for users in the Administrator group.
   - To disable or enable user connections for all applications on a server, enter the name of the server in the Server Selection text box and leave the Application Selection text box blank.
     
     **Note:**
     This also disables connections for users in the Administrator group.
   - To disable or enable user connections for an application, enter the name of the application in the Application Selection text box and leave the Server Selection text box blank.
   - To disable or enable user connections for an application on a server, enter the name of the server in the Server Selection text box and the name of the application in the Application Selection text box.
3. Take one of the following actions:
   - Select Disable Connections, and click OK to confirm.
Select Enable Connections, and click OK to confirm.

Note:
You can also enable connections by clicking in the Enable column of the Disabled Components list.

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.

Note:
Only users assigned the Administrator role can perform a task audit.

The following user activities are logged in the task audit:

- Idle
- Rules Load
- Rules Scan
- Rules Extract
- Consolidation
- Chart Logic
- Translation
- Custom Logic
- Allocate
- Data Load
- Data Extract
- Data Extract via HAL
- Data Entry
- Data Retrieval
- Data Clear
- Data Copy
- Journal Entry
- Journal Retrieval
- Journal Posting
- Journal Unposting
- Journal Template Entry
- Metadata Load
- Metadata Extract
- Member List Load
- Member List Scan
- Member List Extract
- Security Load
- Security Scan
- Security Extract
- Logon
- Logon Failure
- Logoff
- External
- Metadata Scan
- Data Scan
- Extended Analytics Export
- Extended Analytics Schema Delete
- Transactions Load
- Transactions Extract
- Document Attachments
- Document Detachments
- Create Transactions
- Edit Transactions
- Delete Transactions
- Post Transactions
- Unpost Transactions
- Delete Invalid Records
- Data Audit Purged
- Task Audit Purged
- Post All Transactions
- Unpost All Transactions
- Delete All Transactions
- Unmatch All Transactions
- AutoMatch by ID
- AutoMatch by Account
- IC Matching Report by ID
The task audit log includes the following information:

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

**Note:**

The Task Audit feature is available only in the Financial Management Web user interface.

The task audit log information is stored in a table (APPNAME_TASK_AUDIT). You can backup or extract the information in the table. You should monitor the size of the log and clear the log on a regular basis.

To perform a task audit:

1. **Select Administration > Task Audit.**

   **Note:**
   You must have an application open for the Task Audit link to be available.

2. Optional: You can filter the audit by selecting criteria:
   - Enter a start date and an end date.
   - From the Server Selection drop-down list, select a server, or select All.
   - From the User Selection drop-down list, select the user to audit, or select All to audit all users.
   - From the Task Filter drop-down list, select the task to audit, or select All to audit all tasks.

3. Click View.

   The tasks that meet the filtering criteria are displayed. Tasks are sorted from most recent to least recent, as shown in this example.
4 Optional: To export the audit information to a CSV file, click Export, and follow the on screen download instructions.

5 Optional: To delete the entries from the log, click Clear Log to delete the entries from the log.

Note:
When you clear the log, a record of the clear 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 the dimension members of cells.

You can audit Scenario and Account members by setting the EnableDataAudit metadata attribute to Y for the scenarios and accounts to audit. To turn off auditing of Scenario and Account members, change the EnableDataAudit attribute to N.

Note:
Only users assigned to the Administrator role can perform a data audit.

The following user activities are logged in the data audit:

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

The data audit log includes the following information:

- Username
- Activity performed
- Time modified
- Server name
- Point of view
- New value for point of view

Note:
The Data Audit feature is available only in the Financial Management Web user interface. The data audit log information is stored in a table (APPNAME_DATA_AUDIT). You can back up or extract the information in the table. You should monitor the size of the log and clear the log on a regular basis.

➤ To perform a data audit:

1 Select Administration > Data Audit.
   You must have an application open for the Data Audit link to be available.

2 Optional: Set filter options:
   - Enter a start date and an end date to filter by date range.
     Tip: Click \[\text{date} \] to select dates from a pop-up calendar.
   - From the Server Selection drop-down list, select the server to audit, or select All to audit all servers.
   - From the User Selection drop-down list, select the user to audit, or select All to audit all users.
   - Enter member names in the text boxes for the applicable dimensions to filter by dimension members.

Note:
The system filters only by the dimensions for which you specify members. For example, if you specify a member in the Period text box and leave the other text boxes blank, the system filters only by the Period dimension.

3 Click View.
The data changes that meet the filtering criteria are displayed.

4 Optional: In the Point of View column, click a link to view the history for the corresponding cell.

5 Optional: Click Export to export the audit information to a CSV file.

6 Optional: Click Clear Log to delete all entries from the log.

Note:
When you clear the log, a record of the clear remains in the task audit log and cannot be erased.
Monitoring Running Tasks

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

Note:

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

You can view and terminate the following tasks:

- Consolidation
- Data Load
- Extended Analytics Export
- Post All Transactions
- Unpost All Transactions
- Delete All Transactions
- UnMatch All Transactions
- AutoMatch by ID
- AutoMatch by Account
- IC Matching Report by ID
- IC Matching Report by Acct
- IC Transaction Report

You can filter the running tasks by the following task status types:

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

To monitor and stop running tasks:

1. Select Administration > Running Tasks.
You must have an application open for the Running Tasks link to be available.

2 Optional: In the Server Selection text box, enter the server name to filter by application server.
If you leave the Server Selection text box blank, tasks running on all servers are listed.

3 Optional: Set filter options:
- From the User Selection drop-down list, select the user to monitor, or select All to view running tasks for all users.
- From the Task Filter drop-down list, select the task to monitor, or select All to view all running tasks.
- From the Status Filter drop-down list, select the status type to monitor, or select All to monitor all status types.

4 Click View.
The tasks that meet the filtering criteria are displayed.

5 In the Stop Task column, click next to the task to stop.

Note:
Click in the Log column of the list to view the log file for a completed task.

By default, running tasks remain in the database for 900 seconds (15 minutes). You can change the default by modifying the AutoClearDeadTasksAfterSeconds registry key in the HKEY_LOCAL_MACHINE\SOFTWARE\Hyperion Solutions\Hyperion Financial Management\Server\Running Tasks registry location. The minimum value for the registry key is 720 seconds (12 minutes); the maximum value is 864,000 seconds (10 days).

### Scanning For and Deleting Invalid Records

You can use the Delete Invalid Records feature to scan an application for invalid records and to remove them. Administrator access is required to use this feature.

#### Windows Procedure

- To scan for and delete invalid records:
  1 Open the application from which to delete invalid records.
  2 From the navigation frame, select Database Management.
  3 Click the Delete Invalid Records tab.
  4 Browse to or enter the name and path of the log file.
  5 Perform a task:
     - Click Scan.
Web Procedure

To scan for and delete invalid records:
1. Open the application from which to delete invalid records.
2. In Browser View, expand Tasks, and select Data Tasks.
3. Select Database Management.
4. Click Delete Invalid Records.

Note:
This tab is available only to users with Administrator access.

Perform a task:
- Click Scan Only.
- Click Delete Invalid Records.

Changing Settings for Multiple Server Tasks

There are several registry settings that control the amount of consolidations, data loads, and Extended Analytics exports that can run at any one time in a multi-server environment. If you need to run several concurrent consolidations for example, you can change the values in these registry settings.

<table>
<thead>
<tr>
<th>Registry Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxNumConcurrentConsolidations</td>
<td>Controls the number of concurrent consolidations allowed per application server. Any consolidations executed above the value are queued as Scheduled Consolidations. Minimum value is 1; maximum value is 8.</td>
</tr>
<tr>
<td>NumConsolidationThreads</td>
<td>Controls the multi-threading of consolidations per application server. Lowering the value limits the system’s utilization of system resources, resulting in slower consolidation performance. Minimum value is 1; maximum value is 8.</td>
</tr>
<tr>
<td>NumConsolidationsAllowed</td>
<td>Controls the number of consolidations allowed per application across all the application servers. Default value is 8 and the range is 1–20.</td>
</tr>
<tr>
<td>NumDataLoadsAllowed</td>
<td>Controls the number of data loads allowed per application across all the application servers. Default value is 8 and the range is 1–20.</td>
</tr>
<tr>
<td>NumEAExportsAllowed</td>
<td>Controls the number of Extended Analytic exports allowed per application across all the application servers. Default value is 8 and the range is 1–20.</td>
</tr>
</tbody>
</table>

To change a server registry setting:
1. Select Start > Run.
In the Open box, type regedt32, and click OK.

Select HKEY_LOCAL_MACHINE\SOFTWARE\Hyperion Solutions\Hyperion Financial Management\Server\RunningTasks.

Double-click the registry setting that you want to change, change the value, and click OK.

Maintaining Server Logon Information

When you use application server clusters, by default, users are kept on the same application server for subsequent logons. For example, a user logged on to Financial Management who runs reports with Oracle’s Hyperion® Financial Reporting – System 9 will always use the same application server, instead of potentially logging on to different application servers.

You can create the UseStickyServer cluster level registry setting on the client to disable Sticky Server support. You must create the registry setting for each Web server and Win32 client.

➤ To disable UseStickyServer support:

1. Navigate to HKEY_LOCAL_MACHINE\SOFTWARE\Hyperion Solutions\Hyperion Financial Management\Client\Clusters\machinename.

2. Create a REG_DWORD value named UseStickyServer.

3. Enter the value “0”.

Note:

To enable it, change the value to 1, or delete the key. By default, a “1” is assumed if the key is missing.
Managing Application Security

In This Chapter

- Application Security Considerations ................................................................. 56
- Launching the User Management Console from Financial Management .................................. 56
- Loading Application Security ........................................................................ 63
- Extracting Application Security ........................................................................ 68

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

- Load a security file into an application. See “Loading Application Security” on page 63.
- Use the User Management Console to set up security information. See “Launching the User Management Console from Financial Management” on page 56.

Security and access rights enable you to control access to 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.

**Note:**

For information on setting up users, see “Configuring External Authentication for Shared Services” in the Hyperion Security Administration Guide.

The following security roles are required for application administration. For information on assigning roles, see the Hyperion Security Administration Guide.

<table>
<thead>
<tr>
<th>Security Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension Editor</td>
<td>Creates and manages import profiles for dimension creation, as well as creating and managing dimensions manually within the Performance Management Architect user interface or the Classic Application Administration option.</td>
</tr>
<tr>
<td>Security Role</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Application Creator/Financial Management Application Creator</td>
<td>Required to access Classic Application Administration options for Financial Management and Planning using Web navigation. Creates and deploys Performance Management Architect applications. Users with this role can create applications, but can change only the dimensions to which they have access permissions. Required in addition to the Dimension Editor role for Financial Management and Planning users to be able to navigate to their product’s Classic Application Administration options. When a user with Application Creator role deploys an application from Performance Management Architect, that user automatically becomes the application administrator and provisioning manager for that application. The Application Creator can create all applications. 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 Financial Management Configuration Utility.</td>
</tr>
</tbody>
</table>

**Application Security Considerations**

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” on page 63 and “Extracting Application Security” on page 68.

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 User Management Console from Financial Management**

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

1. Create projects. See the Hyperion Security Administration Guide.
2. Create Financial Management applications and add applications to a project. See the *Hyperion Enterprise Performance Management Architect Administrator’s Guide*.

3. Provision users by assigning users and groups to applications and assigning user roles to users and groups. See the *Hyperion Security Administration Guide*.

You can then use the User Management 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 User Management Console from Financial Management:

1. Select **Administration > User Management**.

   The User Management Console opens in the Financial Management browser window.

2. In the **Browser View**, expand **Projects** until you can select the application for which to set up security.

### Assigning Users and Groups to Financial Management Applications

Before you can assign users and groups to applications, you must be assigned the Provisioning Manager role and provision users. See the *Hyperion Security Administration Guide*.

When you select users and groups, only users and groups provisioned for the application are available.

➤ To select users and groups for an application:

1. From **Available Users and Groups**, select an option.

2. Select users and groups to assign to the application, and click ➡️.

   **Tip:**

   Use the Shift and Ctrl keys to select multiple users and groups. Click ➡️ to add all users and groups in the Available Users and Groups column to the Selected Users column. Use ➡️ and ➡️ to remove some or all users and groups from the Selected Users and Groups column.

3. Click **Next** or **Select Classes**.
Setting Up Security Classes for Financial Management Applications

Caution!
The information in this section is provided for use with Classic Financial Management applications only. For information on setting up security using Performance Management Architect, see Hyperion Enterprise Performance Management Architect Administrator’s Guide.

In the Select Classes module, you can perform these procedures:

- “Creating Security Classes” on page 58
- “Deleting Security Classes” on page 58
- “Selecting Security Classes” on page 59

Security classes determine the access that users have to application elements.

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.

Creating Security Classes

➤ To create security classes:

1. In Class Name, enter a name for the security class.

   Note:
The name can contain up to 80 characters.

2. Click Add.

Deleting Security Classes

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 Available Classes, select the security classes to delete.

   **Tip:**
   
   Use the Shift and Ctrl keys to select multiple classes.

2. Click Delete Classes.

3. Click Yes to confirm deletion.

**Selecting Security Classes**

➤ To select security classes for an application:

1. From Available Classes, select the security classes to assign to the application, and click Add Single User or Group.

   **Tip:**
   
   Use the Shift and Ctrl keys to select multiple classes.

2. Click Next or Assign Access.

**Assigning User Access to Security Classes**

After you define users and groups and security classes, you can specify the level of access each user and group has to each security class in the application and set up e-mail alerts.

**Note:**

You must select users and classes for the application before you can access the Assign Access module.

<table>
<thead>
<tr>
<th>Table 3</th>
<th>User Access Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Level</td>
<td>Description</td>
</tr>
<tr>
<td>None</td>
<td>No access to elements assigned to the security class.</td>
</tr>
<tr>
<td>Metadata</td>
<td>View a specified member in a list but cannot view or modify data for the member.</td>
</tr>
<tr>
<td>Read</td>
<td>View data for elements assigned to the security class but cannot promote or reject.</td>
</tr>
<tr>
<td>Promote</td>
<td>View data for elements assigned to the security class and can promote or reject.</td>
</tr>
<tr>
<td>All</td>
<td>Modify data for elements assigned to the security class and can promote and reject.</td>
</tr>
</tbody>
</table>
You can use the Pivot Table feature to toggle between two views for assigning access. For example, if you have users and groups on rows and security classes on columns and click Pivot Table, users and groups will be on columns and security classes on rows.

**Note:**

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

To assign user access to security classes:

1. **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.

2. **From Access, select the access level to assign.**

   **Note:**
   
   See Table 3, “User Access Level,” on page 59.

3. **Click ✓ to apply the level to the selected cells.**

4. **Optional:** To add an e-mail alert, select cells in the table and click Add Alert.

Caution!

The alerting process uses the e-mail addresses stored in the authentication files, such as MSAD, LDAP, or Native Directory. See “Setting Up E-mail Alerting” on page 60.

**Note:**

To remove e-mail alerts, select the cell and click Remove Alert.

5. **Click Save.**

6. **Click Next or Security Report.**

**Setting Up E-mail Alerting**

You can use e-mail alerting for intercompany transactions and during the process management review process. E-mail alerts help highlight a key event or data change in the system. For example, you can send an e-mail 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 e-mail addresses that are stored in the authentication files, such as LDAP, MSAD, or Native Directory.

Process Management Alerting

To set up process management e-mail alerts:

1. For the scenario in the process unit, set the Enable Process Management metadata attribute to “A” to allow alerts.

2. Assign the user to the Receive E-mail Alerts for Process Management role.

Note:
For information on assigning users to roles, see the Hyperion Security Administration Guide.

3. Assign the user to Process Management notifiable roles as defined in Table 4, “Process Management User Roles and Alert Notification,” on page 61.

4. 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.

Users who meet all criteria receive e-mail alerts.

Table 4 Process Management User Roles and Alert Notification

<table>
<thead>
<tr>
<th>Process Unit Level Before or After Action</th>
<th>Process Management User Roles Notified</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Pass</td>
<td>Users with ALL or PROMOTE access to the entity are notified.</td>
</tr>
<tr>
<td>Review Level 1</td>
<td>Reviewer 1 and Submitter roles are notified.</td>
</tr>
<tr>
<td>Review Level 2</td>
<td>Reviewer 2 and Submitter roles are notified.</td>
</tr>
<tr>
<td>Review Level 3</td>
<td>Reviewer 3 and Submitter roles are notified.</td>
</tr>
<tr>
<td>Review Level 4</td>
<td>Reviewer 4 and Submitter roles are notified.</td>
</tr>
<tr>
<td>Review Level 5</td>
<td>Reviewer 5 and Submitter roles are notified.</td>
</tr>
<tr>
<td>Review Level 6</td>
<td>Reviewer 6 and Submitter roles are notified.</td>
</tr>
<tr>
<td>Review Level 7</td>
<td>Reviewer 7 and Submitter roles are notified.</td>
</tr>
<tr>
<td>Review Level 8</td>
<td>Reviewer 8 and Submitter roles are notified.</td>
</tr>
<tr>
<td>Review Level 9</td>
<td>Reviewer 9 and Submitter roles are notified.</td>
</tr>
<tr>
<td>Review Level 10</td>
<td>Reviewer 10 and Submitter roles are notified.</td>
</tr>
<tr>
<td>Submitted</td>
<td>Review Supervisor role is notified. Only users with this role can approve the submitted process unit.</td>
</tr>
<tr>
<td>Approved</td>
<td>Reviewer 1 to Reviewer 10 and Submitter roles are notified.</td>
</tr>
</tbody>
</table>
Process Unit Level Before or After Action | Process Management User Roles Notified
---|---
Published | Users with ALL, READ, or PROMOTE access to the entity are notified.

**Note:**

E-mail alerts are not generated when the process unit is at the Not Started level or for the Sign Off action.

Users with the Application Administrator role do not receive e-mail alerts. For a user with the Application Administrator role to receive e-mail alerts, set up as a separate user and assign the role to receive alerts. The user that performed the action to the process unit is also notified with an e-mail confirmation log stating to whom e-mails were sent. For information on generating e-mail alerts in process management, see the *Hyperion Financial Management — System 9 User’s Guide*.

**Intercompany Transaction Alerting**

➤ To set up intercompany transaction e-mail alerts:

1 Assign the user to the Receive E-mail Alerts for IC Transactions role.

**Note:**

For information on assigning users to roles, see the *Hyperion Security Administration Guide*.

2 Assign the user to the Inter-Company Transaction Admin or Inter-Company Transaction User role.

3 Assign the user ALL, READ, or PROMOTE access to the security classes that are assigned to the scenario and entity in the transaction and add an alert for each security class. See “Assigning User Access to Security Classes” on page 59.

Users who meet all criteria receive e-mail alerts from the Intercompany Transactions or Intercompany Partner Matching Report modules.

For information on generating e-mail alerts in intercompany transactions, see the *Hyperion Financial Management — System 9 User’s Guide*.

**Running Security Reports for Financial Management Applications**

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 CSV file.

➤ To create a security report:

1 **Select a report option:**
   - **Rights** and select options:
Classes by User
Roles by User
Users by Group

2 Select an option:
- Launch Report to open the report in a new window
- Export to File to save the report as a CSV file.

Loading Application Security

Caution!
You can only load security classes for Classic Financial Management applications.

You must load application security before you can load other information to an application.
To remove a role from a user or group you must modify the role in User Management. See the Hyperion Security Administration Guide.

Windows Procedure

➤ To load application security:
1 Open the application.
2 From the navigation frame, select Load Security.
3 For Security Filename, enter the name of the file that to load, or click to find the file.
   Note:
   Security information files use the SEC file extension. You can click View to display the contents of the application security file that you specified.
4 For Log Filename, enter a log file name, or click to find the file.
5 For Delimiter Character, enter 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 Optional: Select Clear All Security Info Before Loading to clear security information for the application before loading 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 Hyperion Security Administration Guide. Before selecting the clear option, review “Clearing and Loading Security Information” on page 65.

7 Select the types of security information to load.

Tip:
Use the Select All and De-Select All buttons to quickly select or de-select the security types.

8 Click Load.

Note:
After you load the security file, you can click the View button next to Log Filename to display the log file.

Web Procedure

Note:
Hyperion recommends that you add Financial Management to the exceptions for your Web pop-up blocker. When you perform some Financial Management tasks, such as loading data on the Web, 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 In Browser View, expand Tasks, and select Load Tasks.

3 Select Load Security.

4 For Security File, enter the name of the file to load, or click Browse to find the file.

Note:
Application security files use the SEC file extension.

5 For Delimiter Character, enter 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 Optional: Select Clear All Security Info Before Loading 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 Hyperion Security Administration Guide. Before selecting the clear option, review “Clearing and Loading Security Information” on page 65.

7 From Security Options, select the types of security information to load.

8 Click Load.

Clearing and Loading Security Information
You can clear security information from an application and then load new security information. For example, if you plan to change security class Class1 to Class2 during the load of security, you need to make the change to all application elements that reference the security class Class1. However, 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 you must perform follow-up steps after you load the new security information.

➢ To clear security information and load a new security file:

1 Extract application elements from the application. See “Before Clearing Security Information” on page 65.

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” on page 66.

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 the following 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 security class information of the journals 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 security class information of the grids 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 security class information of the data forms as necessary.

**After Clearing Security Information**

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

➤ To update metadata after clearing and loading security information, 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 back 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

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

You can extract these types of security information:

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

Note:

Hyperion recommends that you periodically extract security to a back up file. For information on backing up security information, see the Hyperion Shared Services Installation Guide.

Windows Procedure

➤ To extract application security:

1 Open the application.
2 From the navigation frame, select Extract Security.
3 For Security Filename, enter a file name to extract, or click to find the file.

Note:

Application security files use the SEC file extension. You can click the View button next to the security filename to display the contents of the application security file that you specified.

4 For Log Filename, enter a log file name, or click to find the file.
5 For Delimiter Character, enter the character used to separate information in the file.

These characters are valid:

, ~ @ # $ % ^ & | : ; ? \
To extract application security:

1. Open the application.

2. In **Browser View**, expand **Tasks**, and select **Extract Tasks**.

3. Select **Extract Security**.

4. Select the types of security to extract.

5. For **Delimiter Character**, enter 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. **Click Extract**.

7. **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.
You can define metadata for Classic Financial Management applications in two ways:

- Create an XML or APP metadata file and load it to an application.
- Use Metadata Manager to create an XML or APP metadata file and load it to an application.
Note:

You must set up security for an application before you can load metadata. See Chapter 3, “Managing Application Security.”

Sample metadata files are included when you install Financial Management. They are in the Financial Management\Sample Apps directory.

Metadata Attributes

Table 5 lists attributes that you can define for accounts, custom dimensions, entities, and scenarios.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Account</th>
<th>Custom1, Custom2, Custom3, Custom4</th>
<th>Entity</th>
<th>Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccountType</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AllowAdjFromChildren</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>AllowAdjis</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>CalcAttribute</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ConsolidateYTD</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Custom1TopMember, Custom2TopMember, Custom3TopMember, Custom4TopMember</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DefaultFreq</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>DefaultParent</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>DefaultView</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>DefCurrency</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>DeFreqForICTrans</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>EnableCustom1Aggr, EnableCustom2Aggr, EnableCustom3Aggr, EnableCustom4Aggr</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EnableDataAudit</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>HoldingCompany</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>ICPTopMember</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsCalculated</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>IsConsolidated</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 6 lists attributes that you can define for application settings, consolidation methods, and application currencies.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Application Settings</th>
<th>Consolidation Methods</th>
<th>Currencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConsolidationRules</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ConsolMethod</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Control</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Currency</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>DefaultCurrency</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DefaultRateForBalanceAccounts</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>DefaultRateForFlowAccounts</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>DefaultValueForActive</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Application Settings</td>
<td>Consolidation Methods</td>
<td>Currencies</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>----------------------</td>
<td>-----------------------</td>
<td>------------</td>
</tr>
<tr>
<td>Description</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>DisplayInICT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EnableMetadataSecurityFiltering</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICPEntitiesAggregationWeight</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsHoldingMethod</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>MaxCellTextSize</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MaxDocAttachmentSize</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MaxNumDocAttachments</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NodeSecurity</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OrgByPeriodApplication</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PercentConsol</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scale</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>SupportSubmissionPhaseforAccount</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SupportSubmissionPhaseforCustom1,Custom2,Custom3,orCustom4</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SupportSubmissionPhaseforICP</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ToPercentControl</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ToPercentControlComp</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UseSubmissionPhase</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UsedByCalcRoutine</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UsePVAForBalanceAccounts</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UsePVAForFlowAccounts</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UseSecurityForAccounts</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UseSecurityForEntities</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UseSecurityForScenarios</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UseSecurityForCustom1, UseSecurityForCustom2, UseSecurityForCustom3, UseSecurityForCustom4</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UseSecurityForICP</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ValidationAccount</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 7 lists attributes that you can define for system-generated accounts and for the system-generated ICP and Value dimensions.

**Table 7  Metadata Attributes for System-Generated Elements**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>System Accounts</th>
<th>ICP Elements</th>
<th>Value Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SecurityClass</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>NumDecimalPlaces</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>SubmissionGroup</td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

**Defining Accounts**

You define accounts with the attributes described in Table 8.

**Table 8  Account Member Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
</table>
| **AccountType** | (Required) One of the following values:  
  ● ASSET—Store values that represent the assets of a company  
  ● LIABILITY—Store point-in-time balances that represent the liabilities of a company  
  ● REVENUE—Store periodic and year-to-date values that increase net worth if the value is positive  
  Note: In Financial Management releases prior to 4.1, this account type was called Income.  
  ● EXPENSE—Store periodic and year-to-date values that decrease net worth if the value is positive  
  ● FLOW—Store periodic and year-to-date values  
  ● BALANCE—Store unsigned values that relate to a particular point in time  
  ● BALANCERECURRING—Store unsigned values that relate to a particular point in time and that re-occur in future periods  
  ● CURRENCYRATE—Store currency rate information  
  ● GROUPLABEL—Use the account for grouping purposes  
  ● DYNAMIC—Indicates that the account value is calculated dynamically from the data that you are viewing  
  For more information on account types, see "Account Type Behavior" on page 78. |
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
</table>
| CalcAttribute                                              | Description of the calculations in the rules file that are done for this account  
This information is displayed as part of cell info in data forms and data grids. It can contain up to 80 characters including spaces. |
| Custom1TopMember, Custom2TopMember, Custom3TopMember, Custom4TopMember | Which top member in the hierarchy of a Custom dimension is valid for the account  
Only the specified member, including all descendants, is valid for the account. |
| DefaultParent                                               | The default parent for the account                                                                                                                                                                                                                                          |
| Description                                                 | The account description  
The description can contain up to 80 characters, including spaces, and cannot use an ampersand (&).                                                                                                                                 |
| EnableCustom1Aggr, EnableCustom2Aggr, EnableCustom3Aggr, EnableCustom4Aggr | Whether aggregation is enabled for intersections of the Account and Custom dimensions  
This attribute is used for special totals, not summing. Specify Y if the account can aggregate with Custom dimensions or N if it cannot. |
| EnableDataAudit                                             | Whether the account can be audited  
Specify Y to enable auditing of the account or N to disable auditing. The default is N. This attribute, when applied to an account or scenario, determines what can be audited. |
| ICPTopMember                                                | The ICP top member for the account  
The specified member, including all descendants, is valid for the account.                                                                                                                                 |
| IsCalculated                                                | Whether the account is calculated  
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.                                                                                                                                 |
| IsConsolidated                                              | Whether the account is consolidated to a parent account. If the account is not consolidated, it is ignored during consolidation. Specify Y if the account is to be consolidated to a parent account and N if the account is not to be consolidated to a parent account. |
| IsICP                                                       | Specifies whether the account is an intercompany account. If the account is an intercompany account, you must also specify a plug account. Specify one of the following values for this attribute:  
- Y if ICP transactions, including self-ICP transactions, are enabled for the account |
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Attribute</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>N</td>
<td>N if ICP transactions are not enabled for the account</td>
</tr>
<tr>
<td>R</td>
<td>R if ICP transactions are enabled for the account, but the account cannot have ICP transactions with itself</td>
</tr>
<tr>
<td><strong>Member</strong></td>
<td>Specifies the name for the account. This attribute is required. The name must be unique. It can contain up to 20 characters, including spaces, but cannot start with a space.</td>
</tr>
<tr>
<td><strong>NumDecimalPlaces</strong></td>
<td>Specifies the number of digits to be displayed to the right of the decimal point for account values. This attribute is required. Specify a value from 0 to 9 for this attribute.</td>
</tr>
<tr>
<td><strong>PlugAcct</strong></td>
<td>Specifies the name of the account used for identifying discrepancies in intercompany transactions. The PlugAcct attribute is required when the IsICP attribute for the account is selected.</td>
</tr>
<tr>
<td><strong>SecurityClass</strong></td>
<td>Specifies the name of the security class that defines the users who can access the account data. Security class names can contain up to 20 characters. Security access applies only to account data.</td>
</tr>
<tr>
<td><strong>Submission Group</strong></td>
<td>Specifies the submission group. The value can be a number from 0 to 99. The default is blank. A blank value defaults to the value of 1. If you set the submission group to zero (0), the account is not included in the review process.</td>
</tr>
</tbody>
</table>

**Note:** Account names cannot contain spaces if you are using an Oracle database.

Do not use the following characters in an account name:

- Period ( . )
- Plus sign ( + )
- Minus sign ( - )
- Asterisk ( * )
- Slash mark ( / )
- Number sign ( # )
- Comma ( , )
- Semicolon ( ; )
- At sign ( @ )
- Double quote ( " )
- Curly brackets ( { } )
- Ampersand ( & )
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserDefined1, UserDefined2, UserDefined3</td>
<td>Stores custom information for the account. You can enter a maximum of 20 characters. The UserDefined1, UserDefined2, and UserDefined3 functions retrieve the text stored in this attribute.</td>
</tr>
</tbody>
</table>
| UsesLineItems                 | Specifies whether an account can have line items. Specify Y if the account uses line items and N if the account does not use line items. Caution! If you change this attribute after line-item detail is entered, the stored line-item detail may no longer be valid for the account. The following behaviors occur:  
  ● 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.  
  ● 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. |
| XBRL Tags                     | Specifies XBRL tags for the account. You can enter a maximum of 225 characters.                                                                                                                                 |

**Account Type Behavior**

Table 9 describes how account types behave in the system. For example, ASSET accounts do not total across periods, while a REVENUE account provides a year to date total. If you debit an ASSET account, the value that you enter is added to the account. If you credit an ASSET account, the value that you enter is subtracted from the account. All account types, except for GROUPLABEL, contain data.

**Table 9 Account Type Behaviors**

<table>
<thead>
<tr>
<th>Type</th>
<th>YTD Total</th>
<th>Debit</th>
<th>Credit</th>
<th>Default Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASSET</td>
<td>No</td>
<td>Add</td>
<td>Sub</td>
<td>DefaultRateForBalance Accounts</td>
</tr>
<tr>
<td>LIABILITY</td>
<td>No</td>
<td>Sub</td>
<td>Add</td>
<td>DefaultRateForBalance Accounts</td>
</tr>
<tr>
<td>REVENUE</td>
<td>Yes</td>
<td>Sub</td>
<td>Add</td>
<td>DefaultRateForFlow Accounts</td>
</tr>
<tr>
<td>EXPENSE</td>
<td>Yes</td>
<td>Add</td>
<td>Sub</td>
<td>DefaultRateForFlow Accounts</td>
</tr>
<tr>
<td>FLOW</td>
<td>Yes</td>
<td>Add</td>
<td>Sub</td>
<td>None</td>
</tr>
<tr>
<td>BALANCE</td>
<td>No</td>
<td>Add</td>
<td>Sub</td>
<td>None</td>
</tr>
<tr>
<td>BALANCE RECURRING</td>
<td>No</td>
<td>Add</td>
<td>Sub</td>
<td>None</td>
</tr>
</tbody>
</table>
Table 13 indicates how an account type behaves when totaled into a specific type of parent account. 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 account types across the top of the table are identified by the first one or two letters of the account type.

### Table 10  Account Type Behaviors During Aggregation into Parent Accounts

<table>
<thead>
<tr>
<th>Account Type</th>
<th>A</th>
<th>L</th>
<th>R</th>
<th>E</th>
<th>F</th>
<th>B</th>
<th>BR</th>
<th>C</th>
<th>G</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASSET</td>
<td>Add</td>
<td>Sub</td>
<td>Sub</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LIABILITY</td>
<td>Sub</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REVENUE</td>
<td>Sub</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>EXPENSE</td>
<td>Add</td>
<td>Sub</td>
<td>Sub</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FLOW</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>BALANCE</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>BALANCE RECURRING</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CURRENCYRATE</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GROUPLABEL</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DYNAMIC</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

**Note:**

In Table 10, No indicates that the account type is not aggregated into the parent account.

**Example**

The following example illustrates how account types are aggregated into parent accounts:
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:**
   The following account attributes are ignored for dynamic accounts: IsCalculated, IsConsolidated, EnableCustom1Aggr, EnableCustom2Aggr, EnableCustom3Aggr, EnableCustom4Aggr, 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 in “Rule Types” on page 212.

**Defining Custom Members**

You define Custom members by using the attributes in Table 11.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultParent</td>
<td>Specifies the default parent for the Custom dimension member.</td>
</tr>
<tr>
<td>Description</td>
<td>Specifies the description for the custom member. The description can contain up to 80 characters, including spaces, and cannot use an ampersand (&amp;).</td>
</tr>
<tr>
<td>IsCalculated</td>
<td>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 and N if the Custom account is not to be calculated.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Member             | Specifies the name for the custom member. This attribute is required. The name must be unique and can contain up to 20 characters, including spaces, but cannot start with a space.  
**Note:** Custom member names cannot contain spaces if you are using an Oracle database.  
**Note:** The name of a Custom1 dimension member cannot duplicate the name of a consolidation method.  
Do not use the following characters in the custom member name:  
- Period ( . )  
- Plus sign ( + )  
- Minus sign ( - )  
- Asterisk ( * )  
- Slash mark ( / )  
- Number sign ( # )  
- Comma ( , )  
- Semicolon ( ; )  
- At sign ( @ )  
- Double quote ( " )  
- Curly brackets ( { } )  
- Ampersand ( & ) |
| SecurityClass      | Specifies the name of the security class that defines the users who can access the Custom dimension data. Security class names can contain up to 20 characters. Security access applies only to data. |
| Submission Group   | Specifies the submission group. The value can be a number from 0 to 99. The default is blank. A blank value defaults to the value of 1. If you set the submission group to zero (0), the account is not included in the review process. |
| SwitchSignForFlow  | Specifies sign change (Debit or Credit) for FLOW accounts that use the following rules:  
- ASSET to LIABILITY  
- LIABILITY to ASSET  
- EXPENSE to REVENUE  
- REVENUE to EXPENSE  
- BALANCE to FLOW  
- FLOW to BALANCE  
Specify Y if the sign for the account is switched and N if the sign for the account is not switched. |
| SwitchTypeForFlow  | Specifies the account type change for FLOW accounts that use the following rules:  
- ASSET to EXPENSE |
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowAdjFromChildren</td>
<td>Specifies whether journal postings from children are permitted for the parent entity. Specify Y if journal postings from children are permitted and N if journal postings from children are not permitted.</td>
</tr>
<tr>
<td>AllowAdjs</td>
<td>Specifies whether journal postings are permitted for this entity. Specify Y if journal postings are permitted for the entity and N if journal postings are not permitted for the entity.</td>
</tr>
<tr>
<td>DefaultParent</td>
<td>Specifies the default parent for the entity.</td>
</tr>
<tr>
<td>DefCurrency</td>
<td>Specifies the default currency for the entity. This attribute is required.</td>
</tr>
<tr>
<td>Description</td>
<td>Specifies the description for the entity. The description can contain up to 80 characters, including spaces, and cannot use an ampersand (&amp;).</td>
</tr>
<tr>
<td>HoldingCompany</td>
<td>Specifies the holding company for the entity. Can be the name of an entity or &lt;blank&gt;.</td>
</tr>
<tr>
<td>IsICP</td>
<td>Specifies whether the entity is an intercompany entity. Specify Y if the entity is an intercompany entity and N if the entity is not an intercompany entity. If the entity is an intercompany entity, it is displayed in the POV in the ICP dimension under [ICP Entities].</td>
</tr>
<tr>
<td>Member</td>
<td>Specifies the name for the entity. This attribute is required. The name must be unique and can contain up to 20 characters including spaces but cannot start with a space.</td>
</tr>
</tbody>
</table>

**Warning:** Entity names cannot contain spaces if you are using an Oracle database.

Do not use the following characters in the entity name:

- Period ( . )
### Attribute Description

- Plus sign (+)
- Minus sign (-)
- Asterisk (*)
- Slash mark (/)
- Number sign (#)
- Comma (,)
- Semicolon (;)
- At sign (@)
- Double quote ("")
- Curly brackets ({})
- Ampersand (&)

**Note:** You cannot use ALL as the name of an entity.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SecurityAsPartner</td>
<td>Specifies the name of a valid security class for the ICP entity. This attribute enables you to secure the entity in an ICP dimension.</td>
</tr>
<tr>
<td>SecurityClass</td>
<td>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 20 characters.</td>
</tr>
<tr>
<td>UserDefined1, UserDefined2, UserDefined3</td>
<td>Stores custom information for the entity. You can enter a maximum of 20 characters. The UserDefined1, UserDefined2, and UserDefined3 functions retrieve the text stored in this attribute.</td>
</tr>
</tbody>
</table>

### Defining Scenario Members

You define scenario members by using the attributes in Table 13.

#### Table 13  Scenario Member Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConsolidateYTD</td>
<td>Specifies the view for consolidation. This attribute is required. Specify Y for YTD or N for periodic.</td>
</tr>
<tr>
<td>DefaultFreq</td>
<td>Specifies the types of periods for which data input is valid for the scenario. This attribute is required. 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.</td>
</tr>
<tr>
<td>DefaultParent</td>
<td>Specifies the default parent for the scenario.</td>
</tr>
<tr>
<td>DefaultView</td>
<td>Specifies the view to use when &lt;Scenario View&gt; is selected in the point-of-view bar. This attribute is required. Specify YTD or Periodic. 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.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Default view</td>
<td>You must change the extracted line-item detail to match the new default view before loading it.</td>
</tr>
<tr>
<td>DefFreqForICTrans</td>
<td>Specifies the default frequency for intercompany transactions. This attribute must be a valid frequency and can contain a maximum of 20 characters. The default for this attribute is &quot;&lt;blank&gt;&quot;.</td>
</tr>
<tr>
<td>Description</td>
<td>Specifies the description for the scenario. The description can contain up to 80 characters, including spaces, and cannot use an ampersand (&amp;).</td>
</tr>
<tr>
<td>EnableDataAudit</td>
<td>Specifies whether the scenario is audited. This attribute for an account or a scenario determines what can be audited. Specify one of the following values:</td>
</tr>
<tr>
<td></td>
<td>● Y to automatically audit all accounts. Even accounts that have EnableDataAudit set to False will be audited.</td>
</tr>
<tr>
<td></td>
<td>● O to audit only those accounts that have EnableDataAudit set to True.</td>
</tr>
<tr>
<td></td>
<td>● N to disable auditing for all accounts.</td>
</tr>
<tr>
<td>MaximumReviewLevel</td>
<td>Specifies the maximum Process Management review level for the scenario. Specify a review level from 1 to 10. This attribute is required.</td>
</tr>
<tr>
<td>Member</td>
<td>Specifies the name for the scenario. This attribute is required. The name must be unique and can contain up to 20 characters, including spaces, but cannot start with a space.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Scenario names cannot contain spaces if you are using an Oracle database.</td>
</tr>
<tr>
<td></td>
<td>Do not use the following characters in the scenario name:</td>
</tr>
<tr>
<td></td>
<td>● Period ( . )</td>
</tr>
<tr>
<td></td>
<td>● Plus sign ( + )</td>
</tr>
<tr>
<td></td>
<td>● Minus sign ( - )</td>
</tr>
<tr>
<td></td>
<td>● Asterisk ( * )</td>
</tr>
<tr>
<td></td>
<td>● Slash mark ( / )</td>
</tr>
<tr>
<td></td>
<td>● Number sign ( # )</td>
</tr>
<tr>
<td></td>
<td>● Comma ( , )</td>
</tr>
<tr>
<td></td>
<td>● Semicolon ( ; )</td>
</tr>
<tr>
<td></td>
<td>● At sign ( @ )</td>
</tr>
<tr>
<td></td>
<td>● Double quote ( &quot; )</td>
</tr>
<tr>
<td></td>
<td>● Curly brackets ( { } )</td>
</tr>
<tr>
<td></td>
<td>● Ampersand ( &amp; )</td>
</tr>
<tr>
<td>SecurityClass</td>
<td>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 20 characters. For example, a user with None access rights to a scenario can open journal periods for the scenario.</td>
</tr>
<tr>
<td>SupportsProcessManagement</td>
<td>Specifies whether Process Management functionality is enabled. Specify one of the following values:</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| ● Y to enable the Process Management  
● N to disable the Process Management option  
● A to enable Process Management and E-mail alerting. |  |
| UserDefined1, UserDefined2, UserDefined3 | Stores custom information for the scenario. You can enter a maximum of 20 characters. The UserDefined1, UserDefined2, and UserDefined3 functions retrieve the text stored in this attribute. |
| UsesLineItems                              | Specifies whether the scenario can have line items. Specify Y if the scenario can accept line items and N if the scenario cannot accept line items.  
**Note:** If you change this attribute after line-item detail is entered, the stored line item detail may no longer be valid for the scenario. The following behaviors occur:  
● 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.  
● 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. |
| ZeroViewForAdj     | Specifies how to interpret missing, adjusted data values for the period. This attribute is required. Specify YTD or Periodic. |
| ZeroViewForNonadj | Specifies how to interpret missing, nonadjusted data values for the period. This attribute is required. Specify YTD or Periodic. |

**Defining Application Settings**

Application settings apply to an entire 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 Table 14.
Table 14 Application Settings Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConsolidationRules</td>
<td>Specifies whether consolidation rules are supported. Specify one of the following 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.</td>
</tr>
<tr>
<td>DefaultCurrency</td>
<td>Specifies the default currency for the application. This attribute is required.</td>
</tr>
<tr>
<td>DefaultRateForBalanceAccounts</td>
<td>The account that contains the translation rate to use for ASSET or LIABILITY accounts. This attribute is required.</td>
</tr>
<tr>
<td>DefaultRateForFlowAccounts</td>
<td>The account that contains the translation rate to use for REVENUE or EXPENSE accounts. This attribute is required.</td>
</tr>
<tr>
<td>DefaultValueForActive</td>
<td>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.</td>
</tr>
<tr>
<td>EnableMetadataSecurityFiltering</td>
<td>Specifies whether users of an application see all dimension members or only the members to which they have access. The system filters the following dimension members: Scenario, Entity, Intercompany Partner (ICP), Account, Custom1, Custom2, Custom3, Custom4 Specify Y to filter out the dimension members to which the user does not have access. The default for this attribute is N.</td>
</tr>
<tr>
<td>FDM Application Name</td>
<td>Name of the Oracle's Hyperion® Financial Data Quality Management application</td>
</tr>
<tr>
<td>ICPEntitiesAggregationWeight</td>
<td>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.</td>
</tr>
<tr>
<td>MaxCellTextSize</td>
<td>Specifies the maximum number of characters that can be used for cell text. Specify -1 for no limit or...</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>a positive number up to 2,147,483,647. The default value is 8000.</td>
<td></td>
</tr>
<tr>
<td>MaxDocAttachmentSize</td>
<td>Specifies the maximum number of bytes for the size of document attachments. Specify -1 for no limit or a positive number up to 2,147,483,647. The default value is -1.</td>
</tr>
<tr>
<td>MaxNumDocAttachments</td>
<td>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.</td>
</tr>
<tr>
<td>NodeSecurity</td>
<td>Specifies the type of security access for nodes. This attribute is required. Specify Entity to check node data based on security access for the entity and Parent to check node data based on security access for the parent.</td>
</tr>
<tr>
<td>OrgByPeriodApplication</td>
<td>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.</td>
</tr>
<tr>
<td>Support Submission Phase for Account</td>
<td>Specifies whether phased submissions in process management are supported for accounts in the application. Valid values are Y or N. Default is N.</td>
</tr>
<tr>
<td>Support Submission Phase for Custom1, Support Submission Phase for Custom2, Support Submission Phase for Custom3, Support Submission Phase for Custom4</td>
<td>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.</td>
</tr>
<tr>
<td>Support Submission Phase for ICP</td>
<td>Specifies whether phased submissions in process management are supported for ICP members in the application. Valid values are Y or N. Default is N.</td>
</tr>
<tr>
<td>UsePVAForBalanceAccounts</td>
<td>Specifies the default translation method for BALANCE accounts. Specify Y to use the periodic value (PVA) translation method and N to use the value at exchange rate (VAL) translation method.</td>
</tr>
<tr>
<td>UsePVAForFlowAccounts</td>
<td>Specifies the default translation method for FLOW accounts. Specify Y to use the periodic value (PVA) translation method and N to use the value at exchange rate (VAL) translation method.</td>
</tr>
<tr>
<td>UseSecurityForAccounts</td>
<td>Specifies whether accounts in the application are protected by security. Specify Y for security on accounts in the application and N for no security on accounts in the application.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>UseSecurityForCustom1, UseSecurityForCustom2,</td>
<td>Specifies whether custom dimensions in the application are protected by security. Specify Y for security on custom dimensions in the application and N for no security on custom dimensions in the application.</td>
</tr>
<tr>
<td>UseSecurityForCustom3, UseSecurityForCustom4</td>
<td></td>
</tr>
<tr>
<td>UseSecurityForEntities</td>
<td>Specifies whether entities in the application are protected by security. Specify Y for security on entities in the application and N for no security on entities in the application.</td>
</tr>
<tr>
<td>UseSecurityForICP</td>
<td>Specifies whether ICP members in the application are protected by security. Specify Y for security on ICP members in the application and N for no security on ICP members in the application.</td>
</tr>
<tr>
<td>UseSubmissionPhase</td>
<td>Specifies whether phased submissions in process management are used in the application. Valid values are Y or N. Default is N.</td>
</tr>
<tr>
<td>UseSecurityForScenarios</td>
<td></td>
</tr>
<tr>
<td>ValidationAccount</td>
<td>Specifies the name of the account to use for validation. The account used for validation must be an existing account and must have a valid intersection with [ICPTop] in the Custom dimensions.</td>
</tr>
<tr>
<td></td>
<td>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 is used for Submission Phase 1, and Validation Accounts 2 to 9 are used for Submission Phases 2 to 9.</td>
</tr>
</tbody>
</table>

**Organization by Period**

The organization by period functionality enables the most recent consolidation structure to coexist with past structures in the same application.

Organizational structures can change for many reasons, including acquisitions, disposals, mergers, and reorganizations. To support organizational changes, Financial Management uses a system account, Active, to reflect the active or inactive consolidation status of a child into its parent. The Active account acts 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.

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 as an active consolidation member for the current year, scenario, and time period. Children that correspond to ICP members for which the Active account is equal to 0 are considered to be inactive children and are not consolidated. Children that correspond to ICP members for which the Active account is equal to 1 are considered to be...
active children and are consolidated. Active account values can be viewed or changed in Data Explorer. Similarly, changes to active child data affect the parent, while changes to inactive child data does not affect the parent.

The DefaultValueForActive attribute controls the status of children for which the Active account is blank. So, every parent-child intersection does not have to be flagged 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 Table 15.

**Table 15  Consolidation Methods Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
</table>
| ConsolMethod    | Specifies the name for the consolidation method. This attribute is required. The name must be unique and can contain up to 20 characters, including spaces. You cannot use the following characters in the name:  
  - Period ( . )  
  - Plus sign ( + )  
  - Minus sign ( - )  
  - Asterisk ( * )  
  - Slash mark ( / )  
  - Number sign ( # )  
  - Comma ( , )  
  - Semicolon ( ; )  
  - At sign (@)  
  - Double quote ( " )  
  - Curly brackets ( { } )  
  - Ampersand (&) |
| Control         | Specifies the threshold that corresponds to the type of control to be used by the calculation routine. Specify one of the following values for this attribute:  
  - Blank  
  - No  
  - Limited  
  - Full |
<p>| Description     | Specifies the description for the consolidation method. The description can contain up to 80 characters, including spaces, and cannot use an ampersand (&amp;). |
| IsHoldingMethod | 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 and N to use a different method for the holding company. |
| PercentConsol   | Specifies the consolidation percentage applied by the ownership calculation routine. Specify a value for the percent (such as 100) or one of the following keywords: |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>● POWN&lt;br&gt;● POWNMIN</td>
</tr>
<tr>
<td>ToPercentControl</td>
<td>Specifies the upper boundary of the range for PercentControl. Used for the ownership calculation routine. Specify a value between 0 and 100.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> One of the method records must have a value of 100.</td>
</tr>
<tr>
<td>ToPercentControlComp</td>
<td>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 &lt; or &lt;= for this attribute.</td>
</tr>
<tr>
<td>UsedByCalcRoutine</td>
<td>Specifies whether the method is used for the automatic ownership calculation routine. Specify Y to use the method for ownership calculations and N not to use the method for ownership calculations.</td>
</tr>
</tbody>
</table>

**Using Consolidation Methods**

Consolidation methods are defined in the consolidation methods section of the metadata. The methods are used during the consolidation and calculate ownership processes.

When you define consolidation methods in the metadata, the system automatically generates the [ConsolMethod] system list for the Custom1 dimension. The system list consists of all methods defined in the consolidation methods section of the metadata.

There are two ways to assign the consolidation method to an entity for use during consolidation. The method can be assigned manually through data load or data entry. The method can also be assigned by the calculate ownership routine, which is based on the ultimate percent control assigned to the entity.

**Assigning Consolidation Methods Manually**

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

- **POV:** Scenario, Year, Period, View, Entity, Value, Account, C2, C3, C4
- **Scenario:** Applicable scenario
- **Year:** Applicable year
- **Period:** Applicable period
- **Entity:** A parent entity
- **Value:** [None]
- **Account:** [Method], a system generated account called Method
- **C2:** [None]
- **C3:** [None]
C4: [None]

Row: ICP entities (For parents, user can use the system list [ICP Entities] or the user-defined list of selected ICP entities.)

Column: Custom1 (The user should use the system-generated list [ConsolMethods].)

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.

## Defining Currencies

You define currencies for an application by using the attributes in Table 16.

### Table 16  Currency Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
</table>
| Currency       | Specifies the name for the currency. This attribute is required. The name must be unique and can contain up to 20 characters, including spaces. Do not use the following characters in the currency name:  
  ● Period ( . )  
  ● Plus sign ( + )  
  ● Minus sign ( - )  
  ● Asterisk ( * )  
  ● Slash mark ( / )  
  ● Number sign ( # )  
  ● Comma ( , )  
  ● Semicolon ( ; )  
  ● At sign ( @ )  
  ● Double quote ( " )  
  ● Curly brackets ( { } )  
  ● Ampersand ( & ) |
<p>| Description    | Specifies the description for the currency. The description can contain up to 80 characters, including spaces, and cannot use an ampersand (&amp;). |
| DisplayInICT   | Specifies if currencies display in the drop-down list in the intercompany transactions module. Specify Y to display currencies and N to not display currencies. The default is Y. |
| Scale          | 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. Determines how the exchange rate must be entered. Scale is a currency attribute, not an entity attribute. Specify one of the following values for this attribute: |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Blank = None</td>
<td>● 0 = Units</td>
</tr>
<tr>
<td>● 1 = Tens</td>
<td>● 2 = Hundreds</td>
</tr>
<tr>
<td>● 3 = Thousands</td>
<td>● 4 = Ten thousands</td>
</tr>
<tr>
<td>● 5 = Hundred thousands</td>
<td>● 6 = Millions</td>
</tr>
<tr>
<td>● 7 = Ten millions</td>
<td>● 8 = Hundred millions</td>
</tr>
<tr>
<td>● 9 = Billions</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TranslationOperator</th>
<th>Specifies the conversion calculation for intercompany transactions. The default is blank. Specify one of the following values:</th>
</tr>
</thead>
<tbody>
<tr>
<td>● D</td>
<td>to calculate the local currency by dividing the transaction currency by the rate</td>
</tr>
<tr>
<td>● M</td>
<td>to calculate the local currency by multiplying the transaction currency by the rate</td>
</tr>
<tr>
<td>● &lt;blank&gt;</td>
<td>to default the value to D</td>
</tr>
</tbody>
</table>

The system member list [Currencies] is available for the Custom1 and Custom2 dimensions. Currencies that you add to the application are added to the [Currencies] member list. The [Currencies] list enables the entry of currency translation rates for pairs of currencies and provides a way of filtering out non-currency members.

**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**

The system accounts described in Table 17 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 17  System Accounts for Consolidation

<table>
<thead>
<tr>
<th>Account</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>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.</td>
</tr>
</tbody>
</table>
| [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.  
  **Note:** 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

The system accounts described in Table 18 are used for ownership calculations.

**Note:**

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

Table 18  System Accounts for Ownership

<table>
<thead>
<tr>
<th>Account</th>
<th>Description</th>
</tr>
</thead>
</table>
| SharesOwned   | Total number of shares owned. Positive number or 0. Default is 0.  
  **Note:** 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.  
  **Note:** 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.  
  **Note:** Enter the number of shares outstanding, or enter shares outstanding as a percentage. Enter 100 for percentage. |
<p>| VotingOutstanding | Number of voting shares outstanding. A positive number or 0. Default value is 0. |</p>
<table>
<thead>
<tr>
<th>Account</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shares%Owned</td>
<td>Calculated by system.</td>
</tr>
<tr>
<td>Voting%Owned</td>
<td>Calculated by system.</td>
</tr>
</tbody>
</table>

**Editing System-Generated Accounts**

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

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

➤ To modify system accounts:

1. In Metadata Manager, open the 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, and decimal location as needed.
4. As desired, repeat step 3 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 represents a container for all intercompany balances that exist for an account. ICP is a reserved dimension used in combination 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 the following 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 in Metadata Manager. 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. A plug account can be set up as an ICP account. For a plug account to be detailed by ICP, set the IsICP attribute to Y or R so that 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 that the system writes eliminations to [ICP None]. During consolidation, transactions between valid intercompany entities are eliminated. For more information on entity metadata, see “Defining Entity Members” on page 82.

Following are the system-generated ICP elements.

### Table 19 System-Generated ICP Elements

<table>
<thead>
<tr>
<th>ICP Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ICP Top]</td>
<td>Specifies the top intercompany member</td>
</tr>
<tr>
<td>[ICP None]</td>
<td>Specifies that no intercompany member is used</td>
</tr>
<tr>
<td>[ICP Entities]</td>
<td>Specifies the entities that are designated for intercompany transactions</td>
</tr>
</tbody>
</table>

**Editing System-Generated ICP Members**

When you create an application, Intercompany Partner 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. In Metadata Manager, open the 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. As desired, repeat step 2 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, Financial Management automatically creates three Value dimension members for each currency in your application: `CurrencyName`, `CurrencyName Adjs`, and `CurrencyName Total`, where `CurrencyName` is the currency label. For example, for a currency of USD, Financial Management creates the following Value dimension members: USD, USD Adjs, and USD Total.

➤ To modify a Value member description:

1. In Metadata Manager, open the 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. As desired, repeat step 2 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 Custom1 through Custom4 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 give 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

You can use a metadata file of the APP format to add metadata to a Financial Management application. The metadata file sections can be arranged in any order; however, Financial Management automatically processes sections in the following 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 the following order:

- Dimension
- Members
- Hierarchies

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 appear only once in a metadata file.

You can use the following characters as a delimiter:

, ~ @ # $ % ^ | : ; ? \n
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 and 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 version number of the file. 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. The following syntax specifies the file format:

```plaintext
!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. The following examples represent valid file format values:

```plaintext
!FILE_FORMAT = 3.4
!FILE_FORMAT = 3.40
!FILE_FORMAT = 03.40
!FILE_FORMAT = 03.4
```

### Version

This section of a metadata file indicates the version of 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. The following syntax represents the version:

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

The following example represents a valid version value:

```plaintext
!VERSION = 3.5.365
```
Application Settings

This section of a metadata file defines settings that apply to the entire Financial Management application. For information on application settings attributes, see “Defining Application Settings” on page 85.

The following 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
UseSecurityForCustom1=N
UseSecurityForCustom2=N
UseSecurityForCustom3=N
UseSecurityForCustom4=N
UseSecurityForICP=N
EnableMetadataSecurityFiltering=N
```

Currencies

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

```
Label; Scale; Descriptions
```

For more information, see “Defining Currencies” on page 91.

The following example specifies currency attributes:

```
!CURRENCIES
EURO;0;English=European Euro
GBR;0;English=Great Britain Pounds
USD;0;English=United Stated Dollars
```

Dimension

This section of a metadata file defines aliases for Custom dimensions. Create one dimension section for each Custom dimension. Use the following syntax to create the dimension section for a Custom dimension, where N is the number of the Custom dimension:

```
!DIMENSION=CustomN
```

This section contains only the Alias attribute. An alias is an alternate name for a Custom dimension and can have a maximum of 20 characters. Custom dimension aliases are displayed on the Metadata Item Properties tab of the Manage Metadata module.
Enter an alias by using the following syntax, replacing the <> characters with the alias name:

Alias=<>  

The following example specifies a dimension and an alias:

!DIMENSION=Custom2
Alias=Customers

**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 the following syntax:

!MEMBERS=Custom2
'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

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

**Account**

What follows is the syntax for Account dimension members:

'Label, AccountType, IsCalculated, IsConsolidated, IsICP, PlugAcct, Custom1TopMember, Custom2TopMember, Custom3TopMember, Custom4TopMember, NumDecimalPlaces, UsesLineItems, EnableCustom1Aggr, EnableCustom2Aggr, EnableCustom3Aggr, EnableCustom4Aggr, UserDefined1, UserDefined2, UserDefined3, XBRLTags, SecurityClass, ICPTopMember, EnableDataAudit, DefaultParent, Descriptions

For more information on account attributes, see “Defining Accounts” on page 75.

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

!MEMBERS=Account
AdminExpenses;EXPENSE;N;Y;Y; ;Golf;ALL;ALL;2;N;Y;Y;Y; ; 
; ; ; ; ;N;DefaultParent=NetIncome 
CapitalStock;LIABILITY;N;Y; ; ; ; ;ClosingBalance;6;N;Y;Y;Y;Y; 
; ; ; ; ; ;N;DefaultParent=TotalEquity;English=Capital Stock

**Scenario**

What follows is the syntax for Scenario dimension members:

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

For more information on scenario attributes, see “Defining Scenario Members” on page 83.
The following example specifies attributes for two scenarios, Actual and Budget:

!MEMBERS=Scenario
Actual;MTD;Periodic;Periodic;Periodic;N; ; ; ;N; ;
10;Y;N;MTD ;DefaultParent=#root
Budget;MTD;Periodic;Periodic;Periodic;N; ; ; ;Y; ;
10;Y;N;MTD ;DefaultParent=#root

Entity
What follows is the syntax for Entity dimension members:

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

For more information on entity attributes, see “Defining Entity Members” on page 82.
The following example specifies attributes for three entities, California, Canada, and
Connecticut:

!MEMBERS=Entity
California;USD;Y;Y;US;;;;;;DefaultParent=Imbler;English=State of
California;French=California
Canada;USD;Y;N;N;;;;;;;;DefaultParent=Regional
Connecticut;USD;Y;N;US;Northeast;;;;;DefaultParent=Imbler

Custom 1 Through Custom4
Following is the syntax for Custom1 through Custom4 dimension members:

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

For more information on custom attributes, see “Defining Custom Members” on page 80.
The following example specifies attributes for members of the Custom2 dimension:

!MEMBERS=Custom2
[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

Value
You can use the members section to define descriptions for system-defined members of the
Value dimension. In addition, for Value dimension members that the system creates for
currencies, you can define descriptions that will be appended to the currency descriptions. Following is the syntax for Value members:

```
Label;Descriptions
```

You can specify the label of a system-defined Value member. You also can use the following labels to create descriptions that are appended to the descriptions for the corresponding Value members that the system creates for user-defined currencies:

```
<Currency Total>
<Currency Adjs>
<Currency>
```

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

```
[None];English=ValueNone
<Currency Total>;English=Total
<Currency Adjs>;English=Adjs
<Currency>;English=Base
```

Table 20 describes the Value dimension member triplets that the system creates for the USD and Euro currencies.

<table>
<thead>
<tr>
<th>Value Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USD Total</td>
<td>US Dollars Total</td>
</tr>
<tr>
<td>USD Adjs</td>
<td>US Dollars Adjs</td>
</tr>
<tr>
<td>USD</td>
<td>US Dollars Base</td>
</tr>
<tr>
<td>EUR Total</td>
<td>Euro Total</td>
</tr>
<tr>
<td>EUR Adjs</td>
<td>Euro Adjs</td>
</tr>
<tr>
<td>EUR</td>
<td>Euro Base</td>
</tr>
</tbody>
</table>

**Intercompany Partner**

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

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

Following is the syntax for Intercompany Partner members:

```
Label;SecurityClass;Descriptions
```
The following 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.

Following is the syntax for consolidation methods:

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

For more information on consolidation method attributes, see “Defining Consolidation Methods” on page 89.

The following 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 section of the metadata file 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. For more information, see “Custom1 Through Custom4 Hierarchies” on page 105.

To begin a hierarchies section, enter the following 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 Custom2 dimension with the following line:

```
!HIERARCHIES=Custom2
;[None];1
;AllCustomers;0
```
The following topics list the formats for the Hierarchies sections of the Account, Scenario, Entity, and Custom dimensions.

**Account Hierarchies**

Following is the syntax for Account dimension hierarchies:

```
parentmemberlabel;childmemberlabel
```

The following 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**

Following is the syntax for Scenario dimension hierarchies:

```
parentmemberlabel;childmemberlabel
```

The following example specifies Scenario dimension hierarchies:

```
!HIERARCHIES=Scenario
 ; Actual
 ; Budget
```

**Entity Hierarchies**

Following is the syntax for Entity dimension hierarchies:

```
parentmemberlabel;childmemberlabel
```

The following example specifies Entity dimension hierarchies:

```
!HIERARCHIES=Entity
```
Custom1 Through Custom4 Hierarchies

Following is the syntax for Custom1 through Custom4 dimension hierarchies:

```
parentmemberlabel;childmemberlabel;AggregationWeight
```

The following example specifies Custom1 through Custom4 dimension hierarchies:

```
!HIERARCHIES=Custom1
;[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, as described in the following:

- **Value** — Standard members are automatically generated by Financial Management. In addition, after you load metadata, Financial Management 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, Financial Management creates the following triplet of Value dimension members: USD, USD Adjs, and USD Total. You can define descriptions of the system-generated members, as well as descriptions that will be appended to the Value members that the system creates for user-defined currencies.

- **Intercompany Partner** — This dimension is automatically generated by Financial Management. An Intercompany Partner dimension member is generated for each Entity dimension member whose IsICP attribute is set to TRUE. You can define security classes and descriptions for some Intercompany Partner members as described in "Intercompany Partner" on page 102.
Using Metadata Manager Views

You use Metadata Manager to edit and create metadata files. Metadata includes information relating to dimension member attributes and currencies. For example, you can use Metadata Manager to add accounts to an application.

**Note:**

Metadata Manager is available only in the Financial Management Desktop in the Windows client.

Metadata Manager can open files in extensible mark-up language (XML) and APP file formats. You can use Metadata Manager to create an XML or APP file or to edit an XML or APP file that was extracted from a Financial Management application. Metadata files created in Metadata Manager are automatically encoded with the Unicode format, using Little Endian byte ordering. After you create or edit a metadata file, you can load the metadata contained in the file into an application.

**Note:**

All procedures in this topic assume that you have the Metadata Manager workspace open. To open the workspace, select Manage Metadata from the desktop navigation frame.

You cannot edit metadata directly in an application. When you make changes to metadata in Metadata Manager, you are making changes only to the file, not to the metadata in the application. After you extract and modify the metadata in the file, you must load the metadata back into the application for your changes to take effect.

Table 21 describes the views in Metadata Manager.

<table>
<thead>
<tr>
<th>Tab</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tree View</td>
<td>Use a hierarchical view to add or modify members. <strong>Note:</strong> You must use List View for AppSettings, ConsolMethod, and Currencies.</td>
</tr>
<tr>
<td>List View</td>
<td>Use a flat list format to add or modify members.</td>
</tr>
<tr>
<td>Metadata Item Properties</td>
<td>Use to create an alias for a Custom dimension.</td>
</tr>
<tr>
<td>File Properties</td>
<td>Use to create a metadata report.</td>
</tr>
</tbody>
</table>

For most metadata, you can add or modify members in Tree View or List View.

**Note:**

You must use List View for AppSettings, ConsolMethods, and Currencies metadata.
In Tree View, you can add or modify members in a hierarchical view. You add each member as a child or sibling of a member. You enter attributes for each member on the Member Attributes tab. To modify data, select a member and modify it or its attributes. For more information about the Tree View, see “Tree View Tasks” on page 107.

In List View, you add and modify members and member attributes in a flat list. To arrange members in a hierarchy, switch to Tree View and drag the members into the hierarchy. For more information about the List View, see “List View Tasks” on page 113.

**Note:**

Changes made in Tree View are carried over to List View, and vice versa.

### Changing the Metadata File Format

You can convert an APP file to an XML file and vice versa in Metadata Manager by saving the file with the desired file extension.

To save an XML file as an APP file:

1. Click **Open File**, and select the XML file to convert.
2. Click **Save File**.
3. From the **Save as Type** drop-down list, select **APP files (*.app)**.
4. Modify the name of the file, and change the file extension to **APP**.
5. Click **Save**.

**Note:**

When you open a newly-converted APP file in Metadata Manager, you are prompted to enter the delimiter character. Delimiters are required only for APP files.

### Tree View Tasks

All procedures in this topic assume that you have a metadata file open in Metadata Manager. After you make changes to a metadata file, make sure to save the file. For more information, see these procedures:

- “Adding and Modifying Members” on page 108
- “Modifying Node Attributes for Custom Dimensions” on page 109
- “Adding Members from the Member List” on page 109
- “Moving Members” on page 110
- “Promoting Members” on page 110
- “Repositioning Members” on page 110
- “Deleting and Removing Members” on page 111
Adding and Modifying Members

When you add a member to a new hierarchy, you add it as a child of the top member. For example, if you are creating a metadata file and want to set up accounts, the top member, by default, is named Account. You can add child accounts only to Account. Note that you can rename the top member by right-clicking it.

**Note:**

If a new member is a child and the parent exists more than once in the tree, the new child is added as a child of all instances of the parent.

When you add a member to or select a member from Tree View, the attributes of the member are displayed in the right side of the Metadata Manager workspace.

The right side of Tree View contains the additional tabs described in Table 22.

**Table 22  Tree View Tabs**

<table>
<thead>
<tr>
<th>Tab</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member Attributes</td>
<td>Use to display, edit, and enter attributes for the selected member.</td>
</tr>
<tr>
<td>Member List</td>
<td>Use to drag members from the list to the hierarchy.</td>
</tr>
<tr>
<td>Node Attributes</td>
<td>Use for Custom dimensions. This tab contains the attributes applicable to the relationship between the currently selected Custom dimension member and its parent.</td>
</tr>
</tbody>
</table>

➤ To add or modify members:

1. Make sure that the Tree View tab is selected and, from the Metadata Item drop-down list, select a dimension.
2. Do one of the following tasks:
   - To add members, select a member, and click \[\text{add child}\] to add a child to the selected member, or click \[\text{add sibling}\] to add a sibling to the selected member.
   - To modify members, select a member.

**Note:**

You can add children only to the top member in the hierarchy.
3 On the Member Attributes tab, enter or modify attributes for the member.

Note:
For information on attributes, see “Metadata Attributes” on page 72.

4 Repeat steps 2 and 3 until you have added or updated all members.

Modifying Node Attributes for Custom Dimensions

By changing the aggregation weight attribute on the Node Attributes tab, you can modify the relationship between the selected Custom member and its parent. For more information on Custom attributes, see “Defining Custom Members” on page 80.

➤ To modify node attributes:

1 Make sure that the Tree View tab is selected and, from the Metadata Item drop-down list, select a Custom dimension.

2 In the hierarchy, highlight a Custom member.

3 On the right side of the Metadata Manager workspace, select the Node Attributes tab.

4 Modify the AggrWeight attribute.

Adding Members from the Member List

You can add members to the hierarchy by dragging members from the Member List tab and dropping them into the hierarchy structure. If you add a child member to a parent member and the parent exists more than once in the hierarchy, the child member is added as a child member for all instances of the parent member. You can add multiple members simultaneously.

➤ To add members from the Member List tab:

1 Make sure that the Tree View tab is selected and, from the Metadata Item drop-down list, select a dimension.

2 From the Member List tab, select the members to add.
Note:
You can select multiple members by holding down the Ctrl key as you select each member.

3 Drag the selected members to the hierarchy.

Note:
The members are not removed from the Member List tab; they are only copied into the hierarchy.

Moving Members
You can move members in the hierarchy by dragging the members. You can move multiple members simultaneously.

➤ To move members:
1 Make sure that the Tree View tab is selected, from the Metadata Item drop-down list, select a dimension.
2 In the hierarchy, select a member.

Note:
You can select multiple members by holding down the Ctrl key as you select each member.

3 Drag the selected members to the new location.

Promoting Members
You can promote members in the hierarchy by using the promote toolbar button. When you promote a member, it moves it up one level in the hierarchy.

➤ To promote members:
1 Make sure that the Tree View tab is selected and, from the Metadata Item drop-down list, select a dimension.
2 In the hierarchy, select a member to promote.

3 Click.

Repositioning Members
You can reposition members in the hierarchy by using the move up and move down toolbar buttons. Repositioning does not promote members to a new level but moves them up or down within their current level in the hierarchy.

➤ To reposition members in relation to siblings:
1 Make sure that the Tree View tab is selected, and, from the Metadata Item drop-down list, select a dimension.
2 In the hierarchy, select a member to move.

3 Click ‣ or ‣ to reposition the selected member.

Deleting and Removing Members

You can delete members completely from the metadata, or you can remove members from parents and retain the members in the Member List tab. If you remove a member from its parent and the member has no other parent, it is considered an orphaned member in the Member List tab. Deleting a member completely from the hierarchy also deletes it from List View.

➤ To delete or remove members:

1 Make sure that the Tree View tab is selected, and, from the Metadata Item drop-down list, select a dimension.

2 In the hierarchy, select a member.

3 Right-click and select one of the following menu options:
   ● Remove from parent to remove the member from its parent but retain the member in the Member List tab
   ● Delete from dimension to delete the member from the dimension

Adding Orphaned Members

An orphaned member is a member that is not part of the hierarchy and, therefore, does not have a parent or sibling member. When you add members in List View, the members are orphaned until you add them to the hierarchy in Tree View.

Caution!

A metadata file that contains orphaned members cannot be scanned or loaded into an application.

➤ To add orphaned members to the hierarchy:

1 Make sure that the Tree View tab is selected, and, from the Metadata Item drop-down list, select a dimension.

2 On the right side of the Metadata Manager workspace, select the Member List tab.
3 Select Only Show Orphaned Members to view all orphaned members of the dimension.

4 Highlight members and drag them to a new position in the hierarchy.

Removing Orphaned Members

If you select to show only orphaned members, you can easily remove orphaned members from the member list.

Caution!

A metadata file that contains orphaned members cannot be scanned or loaded into an application.

➢ To remove orphaned members:

1 Make sure that the Tree View tab is selected, and, from the Metadata Item drop-down list, select a dimension.

2 On the right side of the Metadata Manager workspace, select the Member List tab.

3 Select Only Show Orphaned Members to display the orphaned members of the selected dimension.

4 Highlight one or more members, right-click, and select Remove Highlighted Members.

Note:

To remove all orphaned members, right-click, and select Select All.

Expanding and Collapsing the Hierarchy

You can expand or collapse the hierarchy by using the expand and collapse toolbar buttons. You can expand or collapse the entire hierarchy, or you can expand or collapse individual parent members.
To expand or collapse the tree:

1. Make sure that the Tree View tab is selected, and, from the Metadata Item drop-down list, select a dimension.
2. In the hierarchy, select a member, and do one of the following tasks:
   - Click to expand the current member and all members below it in the hierarchy.
   - Click to collapse the current member and all members below it in the hierarchy.

### List View Tasks

All procedures in this topic assume that you have a metadata file open in Metadata Manager. After you make a change to a metadata file, make sure to save the file.

**Note:**

After you use List View to add members, use Tree View to add the members to the hierarchy.

For more information, see these procedures:

- “Adding and Modifying Members” on page 113
- “Copying, Cutting, and Pasting Members” on page 114
- “Deleting Members” on page 114
- “Validating Metadata” on page 115
- “Showing or Hiding Columns” on page 115

### Adding and Modifying Members

When you use List View to add members, you can enter the member and its attributes in a flat list. The information that you add or modify is not validated until you change to a different view (for example, Tree View) or manually validate the data.

To add or modify members:

1. Make sure that the List View tab is selected, and, from the Metadata Item drop-down list, select a dimension.
2. Take one of the following actions:
   - To modify a member, select the member and modify it or its attributes.
   - To add a member, click and, in the new line, enter the member name and attributes.

**Note:**

For information on attributes, see “Metadata Attributes” on page 72.

3. Repeat step 2 until you have completed all member modifications and additions.
Copying, Cutting, and Pasting Members

You can cut, copy, and paste members from one cell to another cell or from multiple cells to multiple cells. You can also disable drop-down list cells. Disabling drop-down list cells makes it easier to copy, cut, and paste multiple cells. Changes made in List View are reflected in Tree View.

To copy, cut, or paste members, application settings, or currencies:

1. Make sure that the List View tab is selected, and, from the Metadata Item drop-down list, select a dimension.
2. Optional: Select Disable Combo Boxes.
3. Select a cell or multiple, contiguous cells.
   - Click to cut the information in a cell and store it on the clipboard.
   - Click to copy the information in a cell and store it on the clipboard.
   - Click to paste information from the clipboard to the selected cell.

Deleting Members

When you delete a member you must also delete all attributes associated with the member. Therefore, to delete a member, you must select the entire member row. Changes made in List View are reflected in Tree View.
To delete members:

1. Make sure that the List View tab is selected, and, from the Metadata Item drop-down list, select a dimension.
2. Highlight one or more rows by clicking the row numbers.
3. Click 🗑️

**Validating Metadata**
Changes that you make to members are not validated until you change to a different view (for example, Tree View) or until you select the validate toolbar button. If an error is found during the validation process, the system displays the row number where the error occurred.

**Showing or Hiding Columns**
You can specify the columns of information to be displayed by showing or hiding columns.

- To hide a single column:
  1. Make sure that the List View tab is selected, and, from the Metadata Item drop-down list, select a dimension.
  2. Right-click a column, and select Hide Current Column.

- To show or hide multiple columns:
  1. Make sure that the List View tab is selected, and, from the Metadata Item drop-down list, select a dimension.
  2. Right-click in the grid, and select Show/Hide Columns.
  3. Select the columns to view.
  4. Click OK.

**Sorting Metadata in List View**
You can sort metadata in List View by column. You can sort column information in ascending or descending order.

- To sort a column:
  1. Make sure that the List View tab is selected, and, from the Metadata Item drop-down list, select a dimension.
  2. Double-click a column header to sort the column information.

  **Tip:**
  To sort the column in reverse order, double-click a column header again.
Creating Aliases for Custom Dimensions

All procedures in this topic assume that you have a metadata file open in Metadata Manager. After you make changes to a metadata file, make sure to save the file.

You use the Metadata Item Properties tab to enter aliases for Custom dimensions. An alias describes a Custom dimension.

**Note:**

You can use aliases only for custom dimensions.

To create a dimension alias:

1. Make sure that the **Metadata Item Properties** tab is selected, and, from the **Metadata Item** drop-down list, select a Custom dimension.

2. In the **Alias** text box, enter an alias for the dimension.

Creating Metadata Reports in File Properties

You can apply an XSL style sheet to transform metadata into HTML format for easier viewing on the Web.

Financial Management provides two default report XSL style sheets for metadata reports. For example, for the report to display metadata in hierarchies, use the `HFM_MetadataWithHierarchy.XSL` style sheet.

The style sheets are installed in the Server Working Folder during the installation process. The default location is `C:\Hyperion\FinancialManagement\Server Working Folder \Report Style Sheets\Metadata`. To create your own style sheets instead of using the ones provided, contact your IT administrator for support with XSL.
To create a metadata report:

1. In Metadata Manager, select the File Properties tab, and open a metadata file.

   **Note:**
   By default, metadata files use the XML or APP file extension.

2. Optional: Click View XML to view the metadata before generating the report.

3. In the XSL Style Sheet Filename text box, enter the name of the style sheet to apply to the metadata file; or click ![location](..) to locate a file.

   **Note:**
   By default, metadata report style sheets use the XSL file extension.

4. Click View Metadata to view the formatted report.
Metadata Referential Integrity

To prevent a referential integrity problem from occurring in the application, 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:

When you load metadata, make sure that the Check Integrity option is selected.

Metadata Referential Integrity Checks

The log file provides information about specific changes in metadata attributes that require the system to check existing data in regard to the metadata file that you are loading. Only metadata attributes that have an effect on referential integrity are listed in the table.

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 the following 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 the following format:

Journals::SINGLECA1 Scenario::Actual Year::2007 Value::[Contribution Adjs] Period::January has 1 occurrences of Changed::[SCENARIO::Actual::ZeroViewForAdj: Periodic -> YTD]

This example shows that the metadata integrity error occurs in the SINGLECA1 journal with the following point of view: Scenario Actual, Year 2007, 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.

Loading Metadata

When you load a metadata file, 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.

After you load a metadata file to an application, anyone using the application is notified that the
system has changed and that the user 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.

Load Options
There are several options for how to load metadata to an application. You must select one type
of load method: merge or replace. You have the option to clear all metadata before loading the
new metadata and you have the option to check data integrity. Table 23 describes how to use
each type of load option.

Table 23  Metadata Load Options

<table>
<thead>
<tr>
<th>Load Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Merge</td>
<td>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. 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.</td>
</tr>
<tr>
<td>Replace</td>
<td>All dimension members in the application database are deleted and the members from the load file are put into the database. 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 will be CA with the new info from the load file.</td>
</tr>
<tr>
<td>Clear All Metadata Before Loading</td>
<td>All dimension members and corresponding data, journals, and intercompany transactions in the application database are deleted. Note: If this option is selected it overrides the function of the merge and replace methods.</td>
</tr>
<tr>
<td>Check Integrity</td>
<td>Checks the metadata against the data to ensure integrity. For more information, see &quot;Metadata Referential Integrity” on page 118.</td>
</tr>
</tbody>
</table>
Windows Procedure

To load metadata:

1. Open the application.
2. From the navigation frame, select Load Metadata.
3. In the Metadata Filename text box, enter the name of the file to load; or click ![image] to locate the file.

   **Note:**
   By default, metadata files use the APP or XML file extension.

   You can click View (next to the Metadata Filename text box) to open the metadata file that you selected.

4. In the Log Filename text box, enter a name for the log file; or click ![image] to find the file.
5. If you are loading an APP file, in the Delimiter Character text box, enter the character to be used to separate the metadata in the file.

   Delimiter characters are necessary only for ASCII files that use the APP file extension. Delimiters are not necessary for XML files. The following 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. **Optional:** In the Load Options section, select Clear All Metadata Before Loading.

   **Note:**
   If you select this option, you lose data or journals that are currently active in the application, and you cannot select elements in the Metadata Options section.

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

   It is highly recommended that you select this option as 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. For information on the metadata integrity errors in the log, see “Metadata Log File Referential Integrity Errors” on page 118.
In the **Load Method** section, select one of the following options:

- Merge
- Replace

<table>
<thead>
<tr>
<th>Load Options</th>
<th>Load Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear All Metadata Before Loading</td>
<td>Merge</td>
</tr>
<tr>
<td>Check Integrity</td>
<td>Replace</td>
</tr>
</tbody>
</table>

In the **Metadata Options** section, select the types of metadata to load.

**Tip:**
Use the Select All and De-select All buttons to quickly select or clear all metadata types.

**10 Optional:** Click **Scan** to verify that the file format is correct.

**11 Click** **Load**.

**Note:**
After you load the metadata file, you can click View (next to the Log Filename text box) to display errors encountered during the load process.

**Web Procedure**

**Note:**
Hyperion recommends that you add Financial Management to the exceptions for your Web pop-up blocker. When you perform some Financial Management tasks on the Web 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** In **Browser View**, expand **Tasks**, and select **Load Tasks**.

**3** Select **Load Metadata**.
Note:
This option is only available if you open a Classic Financial Management application.

### Load Metadata

**File Options**

- **Metadata File:** C:\Program Files\Hyperion Solutions
- **Delimiter Character:** ;

**Load Options**

- **Clear all metadata before loading:**
- **Check Integrity**
- **Load Method:** Merge

**Metadata Options**

- **Currency**
- **Custom1**
- **Application Settings**
- **Scenario**
- **Custom2**
- **Consolidation Methods**
- **Entity**
- **Custom3**
- **System Accounts**
- **Account**
- **Custom4**
- **ICPs**
- **Value**

### Instructions

4. In the **Metadata File** text box, enter the name of the file to load; or click **Browse** to find the file.

   **Note:**
   By default, metadata files use the XML or APP file extension.

5. If you are extracting to an APP file, in the **Delimiter Character** text box, enter 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. The following 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. Optional: In the **Load Options** section, select **Clear All Metadata Before Loading**.

   **Note:**
   If you select this option, you cannot select elements in the Metadata Options section.
7  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 as 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. For more information on the metadata integrity errors in the log, see “Metadata Log File Referential Integrity Errors” on page 118.

8  In the Load Options section, select one of the following load methods:
   - Merge
   - Replace

9  In the Metadata Options section, select the types of metadata to load.

Tip:
Use the Select All and De-select All buttons to quickly select or clear all metadata types.

10  Optional: Click Scan to verify that the file format is correct.

11  Click Load.

Extracting Metadata

When you extract metadata, you save the file as an XML or APP file and specify the file name and location. You can extract metadata to view or modify it in Metadata Manager. After you modify metadata, you must load 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.

Windows Procedure

➤ To extract metadata:
   1  Open the application.
   2  From the navigation frame, click Extract Metadata.
   3  In the Metadata Filename text box, enter a name for the file to extract, or click to find the file.
Note:
Metadata files use the APP or XML file extension.

4 In the Log Filename text box, enter a name for the log file, or click to find the file.

5 In the Delimiter Character text box, enter the character to be used to separate the metadata in the file.
Delimiter characters are necessary only for ASCII files using the APP file extension. Delimiter characters are not necessary for XML files. The following characters are valid:

, ~ @ # $ % ^ | : ; ? \n
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 the types of metadata to extract.

Tip:
Use the Select All and De-select All buttons to quickly select or clear all metadata types.

7 Click Extract.

Note:
After you extract metadata, you can click the View button next to the Log Filename text box to display errors encountered during the process.

Web Procedure

To extract metadata:

1 Open the application.

2 In Browser View, expand Tasks, and select Extract Tasks.

3 Select Extract Metadata.
Note:
This option is only available if you open a Classic Financial Management application.

**Extract Metadata**

**File Options**

Delimiter Character: 

- [ ] HFM Application Format

**Metadata Options**

- [ ] Currency
- [ ] Custom1
- [ ] Application Settings
- [ ] Scenario
- [ ] Custom2
- [ ] Consolidation Methods
- [ ] Entity
- [ ] Custom3
- [ ] System Accounts
- [ ] Account
- [ ] Custom4
- [ ] Value

[EXTRACT]

4 In the **Delimiter Character** text box, enter 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. The following 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.

5 **Optional:** Select **HFM Application Format** to extract the file in the APP format.

6 Select the types of metadata to extract.

**Tip:**

Use the Select All and De-select All buttons to quickly select or clear all metadata types.

7 **Click Extract.**

8 **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.
Managing Member Lists

In This Chapter

- Creating Member List Files ................................................................. 127
- Loading Member Lists ........................................................................ 132
- Extracting Member Lists .................................................................... 133
- System Lists by Dimension ................................................................. 134

Member lists enable you to specify a segment of information within a dimension. You create member lists in a VBScript file and load the file into your application. Once loaded, the member lists are available to users of the application. For example, you can select member lists when setting the point of view in Data Explorer or in journals or when copying or clearing data in Database Management. You can create member lists for all Financial Management dimensions.

You can create static and dynamic member lists. Static member lists contain specific 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 have additional accounts in the list, you must add the new accounts to the member list file.

**Note:**

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

A dynamic member list contains the members that, at runtime, meet the criteria that you specify in the member list file. For the Scenario, Year, Period, and Entity dimensions, you can use the current member in the POV as the starting point for the list. Because member lists are dynamic, changes to the metadata are automatically reflected in the list the next time that the list is called.

Sample member list files are included when you install Financial Management. They are located in the `FinancialManagement\Sample Apps` directory.

**Creating Member List Files**

You create member list files using VBScript in an ASCII format supporting multibyte character sets (MBCS), or a file encoded with the Unicode format, using Little Endian byte ordering. Member list files use the LST file extension.
The file format for listing members within a member list file is the same for all dimensions with the exception of Entity. For member lists in the Entity dimension, you must specify the entity and its parent.

You use these subroutines to create member list files:

- Sub **EnumMemberLists ()**: 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.

For information on adding dynamic member lists to the script, see “Dynamic Member Lists” on page 130.

**EnumMemberLists**

Within the EnumMemberLists () subroutine, you use the syntax and functions in the following table to define member lists:

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
</table>
| Dim *Element*Lists(*n*) | Replace *Element* with the dimension for which you are creating member lists and replace *n* with the total number of member lists that you are defining for that dimension. The following syntax states that three member lists are defined for the Entity dimension:  
Dim EntityLists(3) |
| HS.Dimension = "*Element*" | Replace *Element* with the dimension. For example:  
If HS.Dimension = "Entity" Then |
| *Element*Lists(*n*) = "*ListName*" | Replace *Element* with the dimension, replace *n* with a unique identifier for the member list, and replace *ListName* with the name of the member list to use.  
Notice that you can use the @POV keyword to create a dynamic list based on the dimension member that is currently set in the POV. You can use the @POV keyword for the Scenario, Year, Period, and Entity dimensions.  
**Note:** At this time, dynamic POV member lists are only supported when used with Oracle's Hyperion® Financial Reporting – System 9.  
For example:  
EntityLists(1) = "NewEngland"  
EntityLists(2) = "Alloc"  
EntityLists(3) = "AllEntities(@POV)" |
| HS.SetMemberLists *Element*Lists | Replace *Element* with the dimension. For example:  
HS.SetMemberLists EntityLists |

**Example**

What follows is an example of the EnumMemberLists sub-routine. In this example, there are three entity lists defined: NewEngland Alloc which are fixed member lists and AllEntities(@POV) which is a Dynamic POV member list. There is also one Account list defined: ProfitAndLoss.
Sub EnumMemberLists()
    Dim EntityLists(3)
    Dim AccountLists(1)
    If HS.Dimension = "Entity" Then
        EntityLists(1) = "NewEngland"
        EntityLists(2) = "Alloc"
        EntityLists(3) = "AllEntities(@POV)"
        HS.SetMemberLists EntityLists
    Else If HS.Dimension = "Account" Then
        AccountLists(1) = "ProfitAndLoss"
        HS.SetMemberLists AccountLists
    End If
End Sub

**EnumMembersInList**

Within the EnumMembersInList () subroutine, you use the syntax and functions in the following table to define the members of each member list:

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HS.Dimension = &quot;Element&quot;</td>
<td>Replace Element with the dimension. For example: If HS.Dimension = &quot;Entity&quot; Then</td>
</tr>
<tr>
<td>HS.MemberListID</td>
<td>Specifies the member list unique identifier. HS.MemberListID = 1</td>
</tr>
<tr>
<td>HS.AddEntityToList</td>
<td>The member or entity to add to the member list. For Entity lists, use HS.AddEntityToList and list the entity's parent. For all other dimensions, use HS.AddMemberToList. HS.AddEntityToList &quot;UnitedStates&quot;, &quot;Maine&quot; HS.AddMemberToList &quot;July&quot;</td>
</tr>
</tbody>
</table>
| HS Entity.List | Specifies an entity member list. HS.Entity.List (HS.Entity.Member,"[Children]"
| HS.MemberListEntity HS.MemberListScenario HS.MemberListYear HS.MemberListPeriod | Use to specify a dynamic memberlist. |

**Example**

What follows is an example 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

        End If
    End If
End Sub
HS.AddEntityToList "UnitedStates", "Connecticut"
HS.AddEntityToList "UnitedStates", "Massachusetts"
HS.AddEntityToList "UnitedStates", "RhodeIsland"
HS.AddEntityToList "UnitedStates", "Maine"
ElseIf HS.MemberListID = 2 Then
  HS.AddEntityToList "UnitedStates", "Connecticut"
ElseIf HS.MemberListID = 3 Then
  HS.AddEntityToList "UnitedStates", "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"
    HS.AddMemberToList "AdminExpenses"
    HS.AddMemberToList "InterestCharges"
    HS.AddMemberToList "NetIncome"
    HS.AddMemberToList "Taxes"
    HS.AddMemberToList "NetProfit"
  End If
End If
End Sub

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 pull the information from the system. You can use Financial Management functions and arguments to build member lists. Use the Rules Editor to write rules with proper syntax.

The following 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
In this example, the system first gets the list of entities then compares the default currency for the entity value in the array with the desired value of USD. If the value is USD, the entity is added to the list. The system then processes the next entity in the array.

**Dynamic POV Member List**

The bold sections of the following sample show the dynamic POV sections of the member list file.

```vba
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 "","Colombia"
            HS.AddEntityToList "","Germany"
            HS.AddEntityToList "","Spain"
            HS.AddEntityToList "","UK"
        End If
        EntList=HS.Entity.List("","")
        AppCur=HS.AppSettings.Currency
        For each Ent in EntList
            If HS.Entity.DefCurrency(Ent)=AppCur Then
                If HS.MemberListID = 2 Then HS.AddEntityToList "",Ent
                ElseIf Ent<>"[None]" Then
                    If HS.MemberListID = 3 Then HS.AddEntityToList "",Ent
                End If
            End If
        Next
        ScenPOV=HS.MemberListScenario
        YearPOV=HS.MemberListYear
        PerPOV=HS.MemberListPeriod
        EntPOV=HS.MemberListEntity
        If HS.MemberListID = 4 Or HS.MemberListID = 5 Then
            If ( EntPOV <> "" ) Then
                EntList=HS.Node.List("E#" & EntPOV, "[Descendants]", "S#" & ScenPOV & ".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
                        End If
                    Next
                End If
            End If
        End If
    End If
End Sub
```
Loading Member Lists

After you load a member list file into an application, anyone using that application is notified that the system has changed and that the user must log off from the application and log back on.

Windows Procedure

➢ To load member lists:

1. Open the application.
2. From the navigation frame, select Load Member Lists.
3. For Member List Filename, enter the name of the file to load, or click to find the file.

Note:
Member list files use the LST file extension.

You can click View (next to Member List Filename) to open the file that you selected.

4. For Log Filename, enter a log file name, or click to find the file.
5. Optional: Click Scan to verify that the file format is correct.
6. Click Load.

Note:
After you load the file, you can click the View button next to Log Filename to display errors encountered during the load.

Web Procedure

Note:
Hyperion recommends that you add Financial Management to the exceptions for your Web pop-up blocker. When you perform some Financial Management tasks on the Web 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 In the Browser View, expand Tasks and select Load Tasks.
   3 Select Load Member Lists.
   4 For Member Lists File, enter the name of the file to load, or click Browse to find the file.

   Note:
   Member list files use the LST file extension.
   5 Optional: Click Scan to verify that the file format is correct.
   6 Click Load.

   Note:
   If an invalid member list is loaded, Web users may not be able to access the Data Explorer grid. If the following error message is displayed, you may need to correct the member list and load the file:


Extracting Member Lists

Extracted member lists are saved to an ASCII file that supports multibyte character sets (MBCS). 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.

Windows Procedure

➤ To extract member lists:
   1 Open the application.
   2 From the navigation frame, select Extract Member Lists.
   3 For Member List Filename, enter a file name to extract, or click ▼ to find the file.

   Note:
   Member list files use the LST file extension.
   4 For Log Filename, enter the log file name, or click ▼ to find the file.
5 Click **Extract**.

**Note:**
After you extract member lists, you can click the View button to display errors encountered during the extract.

**Web Procedure**

➤ To extract member lists:

1 Open the application.
2 In the **Browser View**, expand **Tasks** and select **Extract Tasks**.
3 Select **Extract Member Lists**.
4 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.

---

**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>
<thead>
<tr>
<th>System List</th>
<th>Scenario</th>
<th>Entity</th>
<th>Account</th>
<th>ICP</th>
<th>Custom</th>
<th>Value</th>
<th>Year</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Hierarchy]</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>[Descendants]</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>[Children]</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>[Base]</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Parents]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Ancestors]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[System]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>[Currencies]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C1, C2</td>
</tr>
<tr>
<td>[ConsolMethod]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C1</td>
</tr>
<tr>
<td>[Inputs]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>[Adjustments]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>[Totals]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>System List</td>
<td>Scenario</td>
<td>Entity</td>
<td>Account</td>
<td>ICP</td>
<td>Custom</td>
<td>Value</td>
<td>Year</td>
<td>Period</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------</td>
<td>--------</td>
<td>---------</td>
<td>-----</td>
<td>--------</td>
<td>-------</td>
<td>------</td>
<td>--------</td>
</tr>
<tr>
<td>[Default Currencies]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[First Generation]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>[Second Generation]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>[Third Generation]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>[Fourth Generation]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>[Fifth Generation]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>[Sixth Generation]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
Many external general ledger systems can generate ASCII text files containing journal information that you can load into a 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 Financial Management and are located in the `FinancialManagement\Sample Apps` directory.

**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. 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:
<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>,</td>
<td>comma</td>
</tr>
<tr>
<td>~</td>
<td>tilde</td>
</tr>
<tr>
<td>@</td>
<td>at sign</td>
</tr>
<tr>
<td>#</td>
<td>number sign</td>
</tr>
<tr>
<td>$</td>
<td>dollar sign</td>
</tr>
<tr>
<td>%</td>
<td>percent sign</td>
</tr>
<tr>
<td>&amp;</td>
<td>ampersand</td>
</tr>
<tr>
<td>^</td>
<td>carat</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>colon</td>
</tr>
<tr>
<td>;</td>
<td>semicolon</td>
</tr>
<tr>
<td>?</td>
<td>question mark</td>
</tr>
<tr>
<td>\</td>
<td>back slash</td>
</tr>
</tbody>
</table>

**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 section of the file 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.

The following example specifies the file format:

```plaintext
!File_Format = 1.0
```

**Version Section**

This section of the file contains the version of Financial Management that you used to journals. The version number is automatically generated when you extract journals.
Note:
This section is not required.

The following example specifies the version:
!Version = 1.1

**Journal Group Section**

This section of the file uses the following syntax to define journal groups.

!GROUP=<journal group>;<journal group description>

For example, the following example defines three journal groups:

!GROUP=Group1;This is the Group1 Journals Group
!GROUP=Group2;This is the Group2 Journals Group
!GROUP=Group3;This is the Group3 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.

The following syntax specifies a standard template:

!STANDARD = <label>, <balancing attribute>, <type>, <journal group>, <securityclass>, <SingleParent.SingleEntity>
!DESC=<journal description>
<parent.entity>, <ICP>, <C1>, <C2>, <C3>, <C4>, <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.

The following syntax specifies a recurring template:

!RECURRING = <label>, <balancing attribute>, <type>, <value>, <journal group>, <securityclass>, <SingleParent.SingleEntity>
!DESC=<journal description>
<parent.entity>, <account>, <ICP>, <C1>, <C2>, <C3>, <C4>, <amount type>, <amount>, <line item desc>
Header Section

This section of the file 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.

The following syntax specifies the scenario, year, and period:

```plaintext
!SCENARIO= Actual
!YEAR = 2007
!PERIOD = July
```

The Header section is followed by journal detail information for Actual, July, 2007.

Journal Subsection

This subsection of the Header section provides journal detail information for the Scenario, Period, and Year specified.

The following table describes the attributes that are used in the Journal sub-section. These attributes are used for regular journals and recurring and standard templates.

### Table 26  Journal Attribute Descriptions

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;label&gt;</td>
<td>User-defined label for journal with up to 20 characters</td>
</tr>
<tr>
<td>&lt;balancing attribute&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● U = unbalanced</td>
</tr>
<tr>
<td></td>
<td>● B = balanced in total</td>
</tr>
<tr>
<td></td>
<td>● E = balanced by entity</td>
</tr>
<tr>
<td>&lt;type&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● R = regular journals</td>
</tr>
<tr>
<td></td>
<td>● A = auto-reversing journals</td>
</tr>
<tr>
<td></td>
<td>● V = auto-reversal journal</td>
</tr>
<tr>
<td>Note:</td>
<td>You cannot load system-generated auto reversals, but you can extract them.</td>
</tr>
<tr>
<td>&lt;status&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● W = Working</td>
</tr>
<tr>
<td></td>
<td>● S = Submitted</td>
</tr>
<tr>
<td></td>
<td>● A = Approved</td>
</tr>
<tr>
<td></td>
<td>● P = Posted</td>
</tr>
<tr>
<td></td>
<td>● R = Rejected</td>
</tr>
<tr>
<td>&lt;value dimension&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● [Contribution Adjs]</td>
</tr>
<tr>
<td></td>
<td>● [Parent Adjs]</td>
</tr>
<tr>
<td></td>
<td>● &lt;Entity Curr Adjs&gt;</td>
</tr>
<tr>
<td></td>
<td>● &lt;Parent Curr Adjs&gt;</td>
</tr>
<tr>
<td>&lt;journal group&gt;</td>
<td>Optional: User-defined parameter with up to 20 characters</td>
</tr>
<tr>
<td>Attribute</td>
<td>Value</td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>Groups need to be preloaded.</td>
</tr>
</tbody>
</table>
| `<security class>` | Optional: Valid security class that is associated with the journal  
**Note:** 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. |
| `<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>` | Description of the journal 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]. |
| `<C1 - C4>` | Optional: Valid members of the Custom1, Custom2, Custom3, and Custom4 dimensions |
| `<amount type>` |  
- `D` = debit  
- `C` = credit |
| `<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 |

The following 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>, <C1>, <C2>, <C3>, <C4>, <amount type>, <amount>, <line item desc>
```

**Loading Journals**

Journals are loaded using the Replace mode, which clears all data for a particular journal label before loading the new journal data.

You can load working, rejected, submitted, approved, and posted journals as well as standard and recurring journal templates. 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. You must manually post the generated reversal.
If you load large journal files on the Web, you should modify IIS settings to increase the AspBuffering limit to avoid errors. You can modify the IIS AspBuffering limit using the Financial Management Configuration Utility. For information, see “Setting ASP Properties” in the Hyperion Financial Management — System 9 Installation Guide.

Windows Procedure

➤ To load journals:
1 Open the application.

   Note:
   Before you can load journals, you must first open the periods to which to load journals. See “Managing Periods” in the Hyperion Financial Management — System 9 User’s Guide.

2 From the navigation frame, select Load Journals.

3 For Journal Filename, enter the name of the file to load, or click to find the file.

   Note:
   Journal files use the JLF file extension.

4 For Log Filename, enter a log file name, or click to find the file.

5 For Delimiter Character, enter the character used to separate the journal 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 Click Load.

   Note:
   After you load journals, you can click the View button next to Log Filename to display errors encountered during the load.

Web Procedure

   Note:
   Hyperion recommends that you add Financial Management to the exceptions for your Web pop-up blocker. When you perform some Financial Management tasks on the Web 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. Open the application.
2. In the Browser View, expand Tasks and select Load Tasks.
3. Select Load Journals.

**Note:**
Before you can load journals, you must first open the periods to which to load journals. See “Managing Periods” in the Hyperion Financial Management — System 9 User’s Guide.

4. For Journals File, enter the name of the file to load, or click Browse to find the file.

**Note:**
Journal files use the JLF file extension.

5. Click Load.

**Note:**
The system displays errors encountered during the load.

### 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). Journal files use the JLF file extension. After you extract journals, you can view and modify the journal information in a text editor.

**Windows Procedure**

To extract journals:

1. Open the application.
2. From the navigation frame, select Extract Journals.
3. For Journal Filename, enter a file name to extract, or click to find the file.

**Note:**
Journal files use the JLF file extension.

4. For Log Filename, enter a log file name, or click to find the file.

5. Select items to extract:
● Templates
● Recurring Templates
● Journals

Note:
If you selected Extract Journals, you must specify the Scenario, Year, and Period from which to extract journals. You can extract journals from a specific period or from all periods.

6 For Delimiter Character, enter the character used to separate the journals data in the file. These characters are valid:
, ~ @ # $ % & ^ | ; ; ? \n
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.

7 Click Extract.

Note:
After you extract journals, you can click the View button to display errors encountered during the extract.

Web Procedure

➤ To extract journals:
1 Open the application.
2 In the Browser View, expand Tasks and select Extract Tasks.
3 Select Extract Journals.
4 Select items to extract:
   ● Templates
   ● Recurring Templates
   ● Journals
5 If you chose to extract Journals, you must specify a Scenario, Year, and one or more Periods from which to extract journals.
6 Optional: Select one or more Entity and Value dimensions to extract.

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.
7 **Optional:** Enter the Label and Group for the journals to extract.

**Note:**
You can use the percent sign ( % ) as a wildcard.

8 Select the check box next to each journal status, journal type, and journal balance type to extract.

9 For **Delimiter Character**, enter the character 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.

10 Click **Extract**.

11 **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.
You create data forms for users who need to enter specific data. You can create data forms in two ways:

- Writing a script in a text file
- Using the Form Builder integrated script editing tool

When you create a data form, you are customizing the data that is displayed. Data forms use the WDF file extension.

**Note:**

To load, extract, and delete Web forms, you must be assigned to the Manage Data Entry Forms role.

**Data Form Script Syntax**

You can use three types of syntax elements when creating data form scripts: keywords, values, and options. Keywords get a line of their own in the script and are placed to the left of the equal sign. Values are placed immediately after the equal sign to complete a line in the script. Options can be added to a line of script where each is delimited by a comma.
**Note:**
When a value is required, it must come before options. Options are never required and can be in any order.

Sample data form scripts are installed with Financial Management and are located in the `Hyperion\FinancialManagement\Sample Apps` directory.

**Note:**
Data entry form script elements are not case-sensitive.

### Table 27  Data Entry Form Script Syntax

<table>
<thead>
<tr>
<th>Script Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddMember</td>
<td>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.</td>
</tr>
<tr>
<td>BackgroundPOV</td>
<td>Use to specify the background dimension members for the form.</td>
</tr>
<tr>
<td>Blank</td>
<td>Use to insert a blank row, column, or cell into the form.</td>
</tr>
<tr>
<td>Cn</td>
<td>Use to define each column in a form.</td>
</tr>
<tr>
<td>CalcByRow</td>
<td>Use to specify if the row calculation is used when a cell has an intersecting column calculation.</td>
</tr>
<tr>
<td>Cell_Link</td>
<td>Use with Link in a row definition to link to another data entry form.</td>
</tr>
<tr>
<td>CellText</td>
<td>Use to display cell text for a row, calculation, or cell.</td>
</tr>
<tr>
<td>CustomHeader</td>
<td>Use to specify custom headers for rows and columns. Use in a row or column definition.</td>
</tr>
<tr>
<td>CustomHeaderStyle</td>
<td>Use to assign custom style attributes to a row or column header.</td>
</tr>
<tr>
<td>DynamicPOV</td>
<td>Deprecated. Do not use.</td>
</tr>
<tr>
<td>FormInputBoxLength</td>
<td>Use to specify the width of the input box on the form.</td>
</tr>
<tr>
<td>FormNumDecimals</td>
<td>Use to specify the number of decimal spaces for the form. This keyword overrides the decimal settings for the currency of the cell. Use NumDecimals to override this setting for a row, column, or cell.</td>
</tr>
<tr>
<td>FormRowHeight</td>
<td>Use to specify the height of all rows in the form.</td>
</tr>
<tr>
<td>FormScale</td>
<td>Use to specify the scale for the form.</td>
</tr>
<tr>
<td>HeaderOption</td>
<td>Use to specify how dimension headers display in the form. Show labels and/or descriptions, set style attributes, set maximum or fixed width.</td>
</tr>
<tr>
<td>Instructions</td>
<td>Use to create instructions in HTML-formatted text and links.</td>
</tr>
<tr>
<td>Script Syntax</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LineItemDetailSinglePeriod</td>
<td>Use to specify if line item detail displays for the selected cell only or for all periods.</td>
</tr>
<tr>
<td>Link</td>
<td>Use with Cell_Link to link to another data entry form.</td>
</tr>
<tr>
<td>MaxCells</td>
<td>Use to specify the maximum number of cells for a data form.</td>
</tr>
<tr>
<td>MaxColsForSparseRetrievalMethod</td>
<td>Use to optimize performance of sparse forms. Use with forms containing more than 10 columns.</td>
</tr>
<tr>
<td>NoSuppress</td>
<td>Use to turn off suppression for one or more rows or columns. This setting overrides other</td>
</tr>
<tr>
<td></td>
<td>suppression settings in the form: SuppressInvalidRows, SuppressNoDataRows, SuppressZeroRows,</td>
</tr>
<tr>
<td></td>
<td>SuppressInvalidColumns, SuppressNoDataColumns, SuppressZeroColumns.</td>
</tr>
<tr>
<td>NumDecimals</td>
<td>Use to specify the number of decimal places for a row, column, or cell. This keyword overrides</td>
</tr>
<tr>
<td></td>
<td>the decimal settings for the cell currency and the decimal setting for the form FormNumDecimals.</td>
</tr>
<tr>
<td>Override</td>
<td>Use to specify a different POV or calculation for one or more rows or columns or to add style</td>
</tr>
<tr>
<td></td>
<td>attributes or set the scale. Use in a row or column definition.</td>
</tr>
<tr>
<td>PrintNumDataColsPerPage</td>
<td>Use to specify the number of columns to print on each page.</td>
</tr>
<tr>
<td>PrintNumRowsPerPage</td>
<td>Use to specify the number of rows to print on each page.</td>
</tr>
<tr>
<td>PrintRepeatHeadersonAllPages</td>
<td>Use to print headers on each page.</td>
</tr>
<tr>
<td>Rn</td>
<td>Use to define each row in a form.</td>
</tr>
<tr>
<td>ReadOnly</td>
<td>Use to specify read only rows, columns, or cells.</td>
</tr>
<tr>
<td>ReportDescription</td>
<td>Use to specify the description for the form.</td>
</tr>
<tr>
<td>ReportLabel</td>
<td>Use to specify the label for the form. Note that you cannot use a period ( . ) in the label</td>
</tr>
<tr>
<td></td>
<td>or a trailing underscore ( _ )</td>
</tr>
<tr>
<td>ReportSecurityClass</td>
<td>Use to specify the security class for the form.</td>
</tr>
<tr>
<td>ReportType</td>
<td>Use to set the form type. The value must be set to WebForm.</td>
</tr>
<tr>
<td>SCalc</td>
<td>Use to specify server-side calculations for a row, column, or cell.</td>
</tr>
<tr>
<td>Scale</td>
<td>Use to specify scale for a row, column, or cell. This setting overrides the scale setting</td>
</tr>
<tr>
<td></td>
<td>of the form FormScale.</td>
</tr>
<tr>
<td>SelectablePOVList</td>
<td>Use to specify the selectable dimension members in the form.</td>
</tr>
<tr>
<td>ShowDescriptions</td>
<td>Use to show descriptions for dimension members.</td>
</tr>
<tr>
<td>ShowLabels</td>
<td>Use to show labels for dimension members.</td>
</tr>
<tr>
<td>String</td>
<td>Use to add a text string to a column, row, or cell.</td>
</tr>
<tr>
<td>Style</td>
<td>Use to specify the style attributes for a row, column, cell, or dimension header.</td>
</tr>
<tr>
<td>Script Syntax</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SuppressColHeaderRepeats</td>
<td>Use to prevent repeated column headers from displaying.</td>
</tr>
<tr>
<td>SuppressInvalidColumns</td>
<td>Use to prevent invalid cells in from displaying in columns.</td>
</tr>
<tr>
<td>SuppressInvalidRows</td>
<td>Use to prevent invalid cells from displaying in rows.</td>
</tr>
<tr>
<td>SuppressNoDataColumns</td>
<td>Use to prevent columns with no data from displaying.</td>
</tr>
<tr>
<td>SuppressNoDataRows</td>
<td>Use to prevent rows with no data from displaying.</td>
</tr>
<tr>
<td>SuppressRowHeaderRepeats</td>
<td>Use to prevent repeated row headers from displaying.</td>
</tr>
<tr>
<td>SuppressZeroColumns</td>
<td>Use to prevent columns with zeros from displaying.</td>
</tr>
<tr>
<td>SuppressZeroRows</td>
<td>Use to prevent rows with zeros from displaying.</td>
</tr>
</tbody>
</table>

**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 Custom1 through Custom4 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 name of the list 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 contains 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.

**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**

```plaintext
C1#Custom1.C2#Custom2.C3#Custom3.C4#Custom4
```

**Table 28 Syntax for BackgroundPOV Keyword**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario</td>
<td>Name of a valid Scenario member.</td>
</tr>
<tr>
<td>View</td>
<td>A valid view.</td>
</tr>
<tr>
<td>Parent</td>
<td>Name of a valid Parent member. This parameter is optional.</td>
</tr>
<tr>
<td>Entity</td>
<td>Name of a valid Entity member.</td>
</tr>
</tbody>
</table>
### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Name of a valid Value member.</td>
</tr>
<tr>
<td>Account</td>
<td>Name of a valid Account member.</td>
</tr>
<tr>
<td>ICP</td>
<td>Name of a valid ICP member.</td>
</tr>
<tr>
<td>Custom1, Custom2, Custom3, Custom4</td>
<td>Name of valid Custom1 through Custom4 members.</td>
</tr>
</tbody>
</table>

**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 dimension member from the user’s point of view is used.

**Example**

```
```

### 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. 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 C1, C2, C3, and so on 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 29 Syntax for Columns Keyword

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>The column number.</td>
</tr>
<tr>
<td>CalcExpression</td>
<td>Use SCalc, Blank, or String. See “SCalc” on page 167, “Blank” on page 152, and “String” on page 170.</td>
</tr>
<tr>
<td>POVExpression</td>
<td>A valid dimension intersection or member list.</td>
</tr>
</tbody>
</table>

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=SCalc(c1+c3),numdecimals:2,CustomHeader:SCalc(c1+c3)
C6=Blank
C7=S#Budget.P#October,Style:font:bold
C8=C1{TotalProducts.[Hierarchy]}

Calc1

This value is deprecated. Use SCalc.

CalcByRow

A keyword used to specify if row calculations are used when a cell has an intersecting column calculation. 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
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. Use with Link. Links are references in row definitions. 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 name of the form 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:1

Example
R1=A#Salaries, CellText:1

**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, the following syntax replaces the three dimension headers in the row with custom headers Scenario, Year, and Month:

R2=S#Actual.Y#1999.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#1999.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
The following example sets the custom header for column 3 to Variance.
C3=SCalc(C1-C2),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” on page 170.

Syntax
This option uses the standards supported by the W3C. See http://www.w3.org/tr/rec-css2.
CustomHeaderStyle:Property:Value

Example
C1=S#Actual.Y#2007,CustomHeaderStyle:;font:italic arial;color:red

DynamicPOV
This keyword is deprecated.
If you have a form that contains the DynamicPOV keyword, take 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 width of the input box. The default is 20 characters wide.

Note:
This keyword does not determine the number of characters that can be entered into the box.

Syntax
FormInputBoxLength=InputLength
Replace InputLength with the number of characters for the width of the input box.
**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 scale setting of the form by using the Scale option within row or column definitions or within a cell override.

**Syntax**

```
FormScale=n
```

Replace \textit{n} with a value from -12 to 12.

**Example**

```
FormScale=0
```

### HeaderOption

Use this keyword to specify options for how dimension headers are displayed. The options currently supported are:

- **Length** - for the maximum length of row headers. This can be a number or can be the word “Fixed”.

  **Note:**
  
  Row headers are truncated with ellipses if they are longer than the specified length.

- **ShowDescription**
- **ShowLabel**
- **Style**

**Syntax**

```
HeaderOptionDimension=Length:n
HeaderOptionDimension=ShowDescription
HeaderOptionDimension=ShowLabel
HeaderOptionDimension=Style:Property:Value;Property:Value...
```

Replace \textit{Dimension} with the name of the row dimension, \textit{n} with the length for the header or “Fixed”, \textit{Property} with the style property and \textit{Value} with the value for the property. For information on Style properties and values, see **Style**.

**Example**

```
HeaderOptionPeriod=Length:4
HeaderOptionScenario=ShowDescription
HeaderOptionsAccount=ShowLabel,Style:Font:italic
```
**Instructions**

Use this keyword to specify the instructions to give to users of the form. 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.<p>Any questions please contact the <A HREF="mailto:bob_smith@hyperion.com">Budget Administrator</A>

**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. Use with **Cell_Link**. Links are references in row definitions. 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 name of the form 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
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 suppression options for the form, 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
```

**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 builder. 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)
```
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| StartCell | An integer value representing the starting point of the override. If the override is defined for a row, this parameter indicates the starting column where the override is applied. In the following example, the override starts with column 2:  
R2=A#Sales, Override(2,3,A#SalesTP) |
| EndCell   | An integer value representing the ending point of the override. In the example above, the override ends with column 3. |
| Override  | 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.  
You can also use these values or options with a POV override or by themselves:  
- Blank  
- NumDecimals  
- ReadOnly  
- SCalc  
- Scale  
- String  
- Style |

Example

In the following override example, the system overrides columns 2 and 3 for row 2 with December as the period and the PriorSales amount instead of the sales amount for the month:  
R2=A#Sales, Override(2,3,Y#2007.P#December.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,SCalc((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)

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

\texttt{PrintNumDataColsPerPage=n}

Replace \textit{n} with the number of columns to print per page.

Example

\texttt{PrintNumColsPerPage=6}

\textbf{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.

\textbf{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

\texttt{PrintNumRowsPerPage=n}

Replace \textit{n} with the number of rows to print per page.

Example

\texttt{PrintNumRowsPerPage=20}

\textbf{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

\texttt{PrintRepeatHeadersonAllPages=Boolean}

Replace \textit{Boolean} with True or False.

Example

\texttt{PrintRepeatHeadersonAllPages=True}
Use this keyword to define a row in the form. The keywords R1, R2, R3, and so on 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**

\[ Rn=\text{CalcExpression} \]
\[ Rn=\text{POVExpression} \]

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>The row number.</td>
</tr>
<tr>
<td>CalcExpression</td>
<td>Use SCalc, Blank, or String. See “SCalc” on page 167, “Blank” on page 152, and “String” on page 170.</td>
</tr>
<tr>
<td>POVExpression</td>
<td>A valid dimension intersection or member list.</td>
</tr>
</tbody>
</table>
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 looks like all other cells but you cannot edit its contents. You can modify the style of a read-only row, column, or cell so that it looks different 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#2007, 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 20 characters. You cannot use these characters in the name:
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. For example: Yes: report_label No: _ No: reportlabel_

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
```

**SCalc**

Use this value to specify calculations in rows, columns, or cells. Use this value within a row or column definition or within a cell override definition. These calculations are performed on the application server.

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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator</td>
<td>The mathematical operator for the calculation. These operators are supported:</td>
</tr>
<tr>
<td></td>
<td>+ - * /</td>
</tr>
<tr>
<td>expression</td>
<td>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. For details and examples, see &quot;Items Supported in SCalc Calculations&quot; on page 167.</td>
</tr>
</tbody>
</table>

**Items Supported in SCalc Calculations**

You can use these types of items in SCalc calculations:

- References to dimension members. The following example references the Account dimension members Purchases and OtherCosts: R6=SCalc(((A#Purchases)-(A#OtherCosts))*100)
- Cell references, using the syntax `Cell(rowIndex,columnIndex)`. The following 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)`. The following 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 \( \text{Col}(\text{columnIndex}) \). The following example adds column 1 and column 3:
  \( C4 = \text{SCalc}(\text{Col}(1) + \text{Col}(3)) \)

- 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, the following SCalc forward reference is allowed:

\[
\begin{align*}
C1 &= \text{A#Sales} \\
C2 &= \text{A#COGS} \\
C3 &= \text{SCalc}(\text{Col}(1) - \text{Col}(2)) \\
C4 &= \text{SCalc}(\text{Col}(3) / \text{Col}(1) \times 100)
\end{align*}
\]

**Scale**

Use this option to specify the scale for uncalculated columns, rows, or cells.

**Syntax**

\( \text{Scale:} n \)

Replace \( n \) with a value from -12 to 12.

**Example**

\( C4 = \text{A#Inventory, Scale:} 2 \)

**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**

\( \text{SelectablePOVList=} \text{Dimension(MemberList)} \)
Table 31  Syntax for SelectablePOVList Keyword

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| Dimension | One of these characters to represent the dimension that is selectable:  
  - S for Scenario  
  - W for View  
  - E for Entity  
  - V for Value  
  - A for Account  
  - I for ICP  
  - C1 for Custom1  
  - C2 for Custom2  
  - C3 for Custom3  
  - C4 for Custom4 |
| Parent    | A valid parent for the dimension. |
| MemberList| Name of a valid member list. |
| SystemList| 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 shown. If there is no description for an entity, only the entity label is shown; 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**

The Style option is interpreted by the browser, not by Financial Management, so it is limited only by what the browser supports. The Style option uses the standards supported by the W3C. See http://www.w3.org/tr/rec-css2. The Style keyword consists of one or more style properties,
separated by semicolons. Each property consists of the property name, a colon, and the value. The value can be one or more words, separated by spaces. For example:

```
Style: color: red; text-align: right
```

**Note:**
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: color: blue; background-color: red; font: bold
```

The following table lists some of the properties and potential values that can be used. For a full list, see the Property Index at http://www.w3.org/TR/REC-CSS2/propidx.html.

**Table 32  Style Examples**

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color (foreground)</td>
<td>The color name or the standard hexadecimal RGB notation. For example:</td>
</tr>
<tr>
<td>Background-color</td>
<td>Style: color: red</td>
</tr>
<tr>
<td></td>
<td>Style: color: rgb(255,0,0)</td>
</tr>
<tr>
<td></td>
<td>Style: background-color: #ff000</td>
</tr>
<tr>
<td></td>
<td>Style: background-color: yellow</td>
</tr>
<tr>
<td></td>
<td>A quick reference for colors can be found at <a href="http://www.cross-browser.com/docs/color_charts.html">http://www.cross-browser.com/docs/color_charts.html</a></td>
</tr>
<tr>
<td>Font-family</td>
<td>The font name. For example:</td>
</tr>
<tr>
<td></td>
<td>Style: font-family: Arial</td>
</tr>
<tr>
<td>Font</td>
<td>The font style. For example:</td>
</tr>
<tr>
<td></td>
<td>Style: font: italic Courier</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> You can combine up to six font properties separated by a space for one Style value. For example:</td>
</tr>
<tr>
<td></td>
<td>Style: font: Arial Italic extra-bold 12pt</td>
</tr>
<tr>
<td>Font-weight</td>
<td>The font weight. Values include demi-bold, demi-light, extra-bold, light, extra-light, demi-light. For example:</td>
</tr>
<tr>
<td></td>
<td>Style: font-weight: extra-bold</td>
</tr>
<tr>
<td>Font-size</td>
<td>The font point size. For example:</td>
</tr>
<tr>
<td></td>
<td>Style: font-size: 12pt</td>
</tr>
<tr>
<td>Alignment</td>
<td>The alignment of the text. Values include left, right, center, and justify. For example:</td>
</tr>
<tr>
<td></td>
<td>Style: text-align: center</td>
</tr>
</tbody>
</table>

You can also mix styles where they intersect by including a semicolon after the Style keyword. 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. To mix the styles from the row and column definitions, you must include the semicolon on the row keyword.

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 also need to 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 mixed by adding a semicolon after the Style keyword in the row definition. See the example for SuppressColHeaderRepeats to see what the data entry form looks like if this syntax is used.

R1=Blank, CustomheaderStyle: Font: Bold, CustomHeader: Balance sheet accounts, Style:; Background-color: pink
C1=S#Actual.Y#@CUR, Style: Background-color: pink; color: green; font: bold

<table>
<thead>
<tr>
<th>Balance sheet accounts</th>
<th>Actual</th>
<th>Budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cash</td>
<td></td>
<td>1,746,137.00</td>
</tr>
<tr>
<td>Short Term Receivables</td>
<td></td>
<td>9,951.32</td>
</tr>
<tr>
<td>Inventories</td>
<td></td>
<td>-10,746.57</td>
</tr>
<tr>
<td>Short Term Investments</td>
<td></td>
<td>2,831.47</td>
</tr>
<tr>
<td>Total Short Term Assets</td>
<td></td>
<td>1,748,173.21</td>
</tr>
<tr>
<td>Computers</td>
<td>33,333.00</td>
<td>13,382.68</td>
</tr>
<tr>
<td>Buildings</td>
<td></td>
<td>4,351.48</td>
</tr>
<tr>
<td>Office Equipment</td>
<td>50,000.00</td>
<td>-948.88</td>
</tr>
<tr>
<td>Transportation</td>
<td>80,000.00</td>
<td>4,719.12</td>
</tr>
<tr>
<td>Fixtures</td>
<td>800,000.00</td>
<td>931.97</td>
</tr>
<tr>
<td>Tangible Assets</td>
<td>963,333.00</td>
<td>22,436.37</td>
</tr>
<tr>
<td>Accumulated Depreciation</td>
<td>22,222.00</td>
<td>-87,002.34</td>
</tr>
</tbody>
</table>

**SuppressColHeaderRepeats**

Use this keyword to specify whether to suppress repeated column headers. The default is True. The following 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.

<table>
<thead>
<tr>
<th></th>
<th>Actual</th>
<th>Budget</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>July</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales</td>
<td>500,000.00</td>
<td>600,000.00</td>
</tr>
<tr>
<td>Purchases</td>
<td>260,000.00</td>
<td>300,000.00</td>
</tr>
</tbody>
</table>
Syntax
SuppressColHeaderRepeats=Boolean
Replace Boolean with True or False.

Example
SuppressColHeaderRepeats=False

**SuppressInvalidColumns**

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
SuppressInvalidColumns=Boolean
Replace Boolean with True to suppress invalid columns or False to display invalid columns.

Example
SuppressInvalidColumns=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
**SuppressNoDataColumns**

Use this keyword to specify whether columns containing nodata cells are suppressed from the form. The default is False.

**Syntax**

SuppressNoDataColumns=Boolean

Replace Boolean with True to suppress columns with no data or False to display columns with no data.

**Example**

SuppressNoDataColumns=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

**SuppressZeroColumns**

Use this keyword to specify whether columns containing zeros are suppressed from the form. The default is False.
Syntax

SuppressZeroColumns=Boolean

Replace Boolean with True to suppress columns with zeros or False to display columns with zeros.

Example

SuppressZeroColumns=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 Periods and Years**

You can use these year and period functions:

- @Cur
- @First
- @Last
- @Next
- @Prev

**Note:**

Year and period values are not case sensitive.

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 2006 December: C1=Y#@Cur.P#@Cur(+1) returns 2007 January.

If the current Point of View is 2006 December: C2=P#@Cur(+1) returns 2007 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 [(+2) for example], 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 the following 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, the following 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.

Creating Data Forms in the Form Builder

Financial Management provides a graphical user interface for creating and maintaining data forms.

For information on the tabs in the Form Builder, see these sections:

- “Properties” on page 178
- “POV” on page 178
- “Columns” on page 179
- “Rows” on page 180
- “Details” on page 180
- “Headers” on page 181
- “Script” on page 181

Some of these tabs provide links to the Member Selector, which you can use to select members and member lists for the form.

To create a data form:

1. Open the application.
2. Select Administration > Manage Documents.
3. Select the Data Entry Forms tab.
4. Click New.
5. Enter the form information on each tab of the Form Builder. You can move between tabs by clicking the tab or by clicking Back and Next at the bottom of the Form Builder. Each tab also contains these buttons:
   - **Scan** - to scan the form for proper syntax.
     After the system scans the form, it displays a page containing information such as the start and end times of the scan. The page also includes one of the following:
     - If the form is valid, the page includes a **Preview** button, which you can use to view the form.
     - If the form contains errors, the page lists the errors.
   - **Update** - to post changes to the server.
   - **Reset** - to reset form values. This button can be useful if you make a mistake.
6. Click **Save** at the bottom of the Form Builder, then select a file name and location.
After the system attempts to save the form, it displays a page containing information such as the start and end times of the save operation. Whether the form is saved and the rest of the page content depends upon the following:

- If the form is valid, the system saves the form and the page includes a Preview button, which you can use to view the form.
- If the form contains errors, the form is not saved and the page lists the errors.

*Note:*

If you attempt to close the form without saving the script, the system warns you and provides the option to save before continuing.

**Properties**

The Properties tab specifies general information for the data form. The following table describes the information on the Properties tab:

<table>
<thead>
<tr>
<th>Text Box</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Form name. See ReportLabel.</td>
</tr>
<tr>
<td>Security Class</td>
<td>Form security class. See ReportSecurityClass.</td>
</tr>
<tr>
<td>Description</td>
<td>Form description. See ReportDescription.</td>
</tr>
<tr>
<td>Instructions</td>
<td>Instructions that administrators provide for form users. See Instructions.</td>
</tr>
</tbody>
</table>

If you hover over a label, a tooltip displays the script syntax.

**POV**

The POV tab specifies the Background and Selectable POV:

- A Background POV for the form specifies the initial members for dimensions. Use the Member text boxes to specify the Background POV members. Members specified in the Member text boxes are added to the BackgroundPOV keyword, which is described in “BackgroundPOV” on page 151.
- A Selectable POV for the form consists of the dimensions for which users can select members. You use the Selectable text boxes to specify the member lists from which users can select members. Members specified in the Selectable text boxes are added to the SelectablePOVList keyword. For information on the keyword and the syntax used for member lists, see “SelectablePOVList” on page 168.

You can specify a Background or Selectable POV for a dimension in two ways:

- Type a member label or the syntax for a member list in the applicable text box.
● Click for the applicable text box to select members and member lists with the Member Selector.

These rules apply to using the POV tab:

● If Member and Selectable are blank 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.

**Columns**

The Columns tab specifies the dimension members to display in the columns of the form. Each text box in the Col section represents a column. You can specify the members of a column in two ways:

● Type the member label in the text box for the column.

● Click for the column to select members with the Member Selector.

If you have filled in all of the currently visible column text boxes, you can add a new text box by pressing the Tab key. You can define a maximum of 24 columns in the Columns tab.

*Note:*

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” on page 152.

Dimension elements specified in the column section override elements that are set in the Background POV or Selectable POV.

*Note:*

You can specify one member list per column.

As you enter information in each column, the Options section for that column is available at the bottom of the Form Builder.

If you hover over a label, a tooltip displays the script syntax. For valid values for each option, see “Data Form Script Syntax” on page 147.

*Note:*

You can also use Other to specify syntax for column keywords not exposed in the Options section, such as Blank and ReadOnly. In addition, if the definition of a column in the script contains unexposed syntax, or deprecated or invalid syntax, that syntax is displayed in Other.
**Rows**

The Rows tab specifies the dimensions to display in the rows of a form. Each text box in the Row section represents a row. You can specify the members of a row in two ways:

- Type the member label or member list syntax in the text box for the row.
- Click ![Member Selector](image) for the rows text box to select members and member lists with the Member Selector.

If you have filled in all of the currently visible Row text boxes, you can add a new text box by pressing the Tab key. You can define a maximum of 100 rows in the Rows tab.

**Note:**

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” on page 164.

Dimension elements specified in the row section override elements set in the Background POV or Selectable POV.

As you enter information in each row, the options section for that row is available at the bottom of the Form Builder.

If you hover over a label, a tooltip displays the script syntax. For information on the valid values for each option, see “Data Form Script Syntax” on page 147.

**Note:**

You can also use Other to specify syntax for row keywords not exposed in the Options section, such as Blank and ReadOnly. In addition, if the definition of a row in the script contains unexposed syntax, or deprecated or invalid syntax, that syntax is displayed in Other.

**Details**

The Details tab specifies grid, print, display, and suppression options for the Data Entry form.

**Note:**

The items in the Details tab 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.

If you hover over a label, a tooltip displays the script syntax. For valid values for each option, see “Data Form Script Syntax” on page 147.
Headers

The Headers tab enables you to specify the maximum length of the labels or descriptions for members and member lists. You can specify different lengths for different dimensions.

The Headers tab also contains the Other text box, which has two uses:

- If syntax is added for the `HeaderOption` keyword but the syntax is not exposed through a new control in the Headers tab, 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.

Script

The Script tab enables you to view, modify, and print the script. You enter the syntax in the text box on the Script tab. The Script tab also contains these buttons:

- Print - opens a new browser window that contains a printer-friendly version of the syntax.
- More - increases the height of the text box that contains the syntax. When you click More, the button is replaced by the Less button.
- Less - decreases the height of the text box that contains the syntax. When you click Less, the button is replaced by the More button.

Editing Data Forms

You can edit a data form using the Form Builder.

**Note:**

After you edit a data form, you must load the updated script to the application to see the changes that you made.

➤ To edit a data form:

1. Open the application.
2. Select Administration > Manage Documents.
3. Select the Data Entry Forms tab.
4. To edit a form do one of these tasks:
   - Select the check box next to a form and click Edit.
   - Click the edit icon next to a form.

**Note:**

You can select only one form at a time to edit.
5 Edit the form information on each tab of the Form Builder. You can move between tabs by clicking the tab or by clicking Back and Next at the bottom of the Form Builder.

Note:
To post changes to the server, click Update.

Tip:
If you make a mistake, click Reset at the bottom of the Form Builder to reset values.

6 To scan the form for proper syntax, click Scan at the bottom of the Form Builder.

7 Click Save at the bottom of the Form Builder.

8 Select a file name and location.

Creating Data Form Folders

You can create a folder hierarchy for data forms. A Root folder is available by default and cannot be deleted. New folders are created under the Root folder. Organizing data forms into folders may help if you have many forms.

➤ To create folders for data forms:
1 Open the application.
2 Select Administration > Manage Documents.
3 Select the Data Entry Forms tab.
4 Select New Folder.
5 Enter a folder name.
   The name can contain a maximum of 20 characters, including spaces.
6 Optional: Enter a folder description.
   The description can contain a maximum of 40 characters, including spaces.
7 From Security Class, select the security class to assign to the folder.

   Note:
   Users of this folder must have access rights to the specified security class.
8 Click OK.
Viewing and Hiding Private Data Forms

You can view private documents of other users. This feature is available only to the application administrator. To view only private documents, click Show Private Documents; to view only public documents, click Hide Private Documents.

Loading Data Forms

After you create data form scripts, you load them into an application. For information on creating a data form script, see “Data Form Script Syntax” on page 147.

Note:

Hyperion recommends that you add Financial Management to the exceptions for your Web pop-up blocker. When you perform some Financial Management tasks on the Web 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 Open the application.
2 Select Administration > Manage Documents.
3 Select the Data Entry Forms tab.
4 Click Load.
5 Click Browse and locate the data form script to load, then click Open.

Note:

Data form scripts use the WDF file extension.

6 Optional. Perform one of these steps:
   • To override the security class specified in the file being loaded, select Override, then select the security class from Security Class.
   • To make this form available only to you, select Private. If you select this check box, the Override check box and Security Class drop-down list are not available.

7 Repeat step 5 and step 6 until you have added the forms to load.
8 Optional: Select Overwrite Existing Documents if you are updating data forms.
9 Click OK.

Note:

All data forms that you load must include a valid ReportLabel. If you have 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 load it to the application. Make sure to overwrite the old file when you load the script. You can also use the model for a new script.

Note:
If you are loading a script to 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 Administration > Manage Documents.
3 Select the Data Entry Forms tab.
4 Click Extract.
5 Click the file name link for the file to extract.
6 Click Save.
7 Specify the name and location for the extracted file, or leave the default name and path.

Note:
Data forms use the WDF file extension.
8 Click Save.
9 Repeat steps 5-8 until you have extracted all data forms that you selected in step 3.
10 Click OK.

Deleting Data Forms

To be able to delete data forms, you must be an administrator with the security role of Manage Data Entry Forms.

➤ To delete data forms:
1 Open the application.
2 Select Administration > Manage Documents.
3 Select the Data Entry Forms tab.
4 Select the check box next to the data form to delete.
Note:
You must delete data forms from a folder before you can delete a folder.

5 Click Delete.
An Extended Analytics star schema enables you to use Oracle's Hyperion® Essbase® – System 9 to analyze data and produce reports. To do so, you use the star schema to send data to an Oracle's Hyperion® Essbase® – System 9 database.

Note:
You create the database after you export to the star schema. See the *Database Administrator’s Guide* for your release.

You can also use a star schema with third-party products.

To use the Extended Analytics module, you must set up an OLAP data source name (DSN) for the database that will store star schemas, then configure the DSN with the Financial Management configuration utility. For instructions, see the *Hyperion Financial Management — System 9 Installation Guide*.

Note:
To store star schemas in multiple databases you can create a DSN for each database.

To use Extended Analytics, you must be assigned the security role of Administrator or Extended Analytics.
Extended Analytics Registry Settings

The following registry keys can be set in the HKEY_LOCAL_MACHINE\SOFTWARE\Hyperion Solutions\Hyperion Financial Management\Server registry location.

Table 34  Extended Analytics Registry Settings

<table>
<thead>
<tr>
<th>Entry Name</th>
<th>Data Type</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EACommandTimeout</td>
<td>REG_DWORD</td>
<td>0</td>
<td>The number of seconds that must elapse before a SQL query is determined to be deadlocked. Specify a value from 0, which means no time out, to 65535.</td>
</tr>
<tr>
<td>NumEAThreads</td>
<td>REG_DWORD</td>
<td>8</td>
<td>The number of threads used during processing of data. Specify a value from 1 to 32.</td>
</tr>
<tr>
<td>EAIgnoreInvalidMemberIDs</td>
<td>REG_DWORD</td>
<td>0</td>
<td>Determines whether to fail if an invalid dimension member ID is encountered. Specify 1 to ignore invalid IDs. Specify 0 to have the system fail if an invalid ID is encountered.</td>
</tr>
<tr>
<td>EAUseSQLBinding</td>
<td>REG_DWORD</td>
<td>1</td>
<td>Determines whether to use SQL binding. Specify 1 to use SQL binding or 0 to disable SQL binding. <strong>Caution:</strong> If you disable SQL binding, performance may be adversely affected.</td>
</tr>
<tr>
<td>ConnectionPoolingTrace</td>
<td>REG_DWORD</td>
<td>0</td>
<td>Determines whether to trace connection pool activity to log. Specify 1 to trace or 0 to disable trace.</td>
</tr>
<tr>
<td>EATraceADOErrors</td>
<td>REG_DWORD</td>
<td>0</td>
<td>Determines whether to trace ActiveX Data Object (ADO) activity to log. Specify 1 to trace or 0 to disable trace. This setting is not applicable to ODBC.</td>
</tr>
<tr>
<td>EAUseODBC</td>
<td>REG_DWORD</td>
<td>1</td>
<td>Determines whether to use ODBC to interact with the external data store. Specify 1 to use ODBC or specify 0 to use ADO.</td>
</tr>
<tr>
<td>EALogEntirePOV</td>
<td>REG_DWORD</td>
<td>0</td>
<td>Determines whether to log the pov being captured in the Extended Analytics dump to the log file (in the temp directory). Specify 1 to log the POV or 0 to disable the logging.</td>
</tr>
</tbody>
</table>

Star Schema

The Extended Analytics module enables you to 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 the following data:

- Scenario - Actual
- Year - 2007
- Period - July
- 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's Essbase® Integration Services to create one or many data cubes to reflect the audience that needs to see and use the Budget information. The star schema will contain 180 data combinations for these members (1 * 1 * 1 * 4 * 1 * 3 * 1 * 3 * 5 * 1 * 1 = 180).

Star Schema Formats

You select one of these aggregation options when you create a star schema:

- Standard
- Metadata Only
- Selected Metadata Only
- Essbase
- Data Warehouse
The aggregation option that you select determines the schema format used. Each schema format generates a different set of tables. There are three schema formats:

- Standard Essbase Schema - this schema is used for the Standard, Metadata Only, and Selected Metadata Only aggregation types.
- SQL and Essbase Aggregation Schema - this schema is used for the Essbase aggregation type.
- Warehouse Normalized Hierarchy Schema - this schema is used for the Data Warehouse aggregation type.

For each schema format, the system creates a `<PREFIX>_FACT` table that contains keys to the 13 dimension tables and one data field. The system also creates `<PREFIX>_DIMENSION` tables, a HFM_EA_EXTRACT table to track extract timestamps for metadata, and a `<PREFIX>_LOCK_ACCESS` table to track writer and reader locks. A writer lock is used when metadata is being changed, for example when the create or replace process is being used. A reader lock is used when metadata is not being changed and the system is updating data in the FACT table, for example when an update process is being used. The rules for locking are:

- Only one writer at a time can execute against the same prefix on the same DBMS instance.
- If a writer is executing or is in the queue, no readers can execute until the writer has completed.
- Multiple readers can execute simultaneously regardless of the point of view.

**Note:**

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 aggregation type:

- HFM_EA_EXTRACT1
- HFM_LOCK_ACCESS1
- 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_CUSTOM1 and DEMO_CUSTOM1_BASE
DEMO_CUSTOM2 and DEMO_CUSTOM2_BASE
DEMO_CUSTOM3 and DEMO_CUSTOM3_BASE
DEMO_CUSTOM4 and DEMO_CUSTOM4_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 Financial Management database as the destination database for the data export.

Note:

For Oracle 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.

Hyperion recommends that you add Financial Management to the exceptions for your Web pop-up blocker. When you perform some Financial Management tasks on the Web 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 export Financial Management data to it:

1 Open the application.
2 Select Administration > Extended Analytics.
3 Do one of these tasks:
   - From Template, select the saved POV template to use.
   - Select the POV dimension members to export.
4 From Destination Database (DSN), select the database to which you are exporting the star schema.
Note:
Do not use the same Financial Management database that you are exporting data from as the destination database.

5 For Relational Table Prefix, enter the prefix that will identify the tables for the star schema.

Note:
The prefix can contain up to 10 alpha-numeric characters and must start with a letter. The default prefix is the name of the application.

6 From Aggregation Option, select the type of aggregation to perform on the data.
   ● Standard
   ● Metadata Only - extracts metadata only
   ● Selected Metadata Only - extracts only the metadata for the selected dimension members.
   ● Essbase
   ● Data Warehouse

7 Optional: Uncheck the Exclude Dynamic Accounts option to include dynamic accounts in the star schema.

8 Select Create Star Schema.

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 clears data from the Fact table only.

Hyperion recommends that you add Financial Management to the exceptions for your Web pop-up blocker. When you perform some Financial Management tasks on the Web 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 Open the application.
2 Select Administration > Extended Analytics.
3 Do one of these tasks:
   ● From Template, select the saved POV template to use.
   ● Select the POV dimension members to export.
4 From Destination Database (DSN), select the database that contains the star schema.
5 For **Relational Table Prefix**, type the prefix that identifies the tables for the star schema.

**Note:**

The prefix can contain up to 10 alpha-numeric characters and must start with a letter. The default prefix is the name of the application.

6 From **Aggregation**, select the type of aggregation to perform on the data.
   - Standard
   - Metadata Only - extracts metadata only
   - Selected Metadata Only - extracts only the metadata for the selected dimension members.
   - Essbase
   - Data Warehouse

7 **Optional:** Uncheck the **Exclude Dynamic Accounts** option to include dynamic accounts in the star schema.

8 Select **Update Star Schema**.

---

**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 Open the application.
2 Select **Administration > Extended Analytics**.
3 From **Destination Database (DSN)**, select the database that contains 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 alpha-numeric characters and must start with a letter. The default prefix is the name of the application.

5 **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 Open the application.
2 Select Administration > Extended Analytics.
3 Select the POV dimension members.
4 From Destination Database (DSN), select the database to which you are exporting the star schema.
5 For Relational Table Prefix, type the prefix that identifies the star schema’s tables.

Note:

The prefix can contain up to 10 alpha-numeric characters and must start with a letter. The default prefix is the name of the application.

6 Click ✅.
7 Type a template name.
8 Optional: To overwrite a template, select Overwrite.
9 Click Save.
10 Click OK.

Deleting a Star Schema Template

➤ To delete a star schema template:
1 Open the application.
2 Select Administration > Extended Analytics.
3 From Template, select the template to delete.
4 Click ✅.
5 Click OK to confirm.
6 Click OK.
Defining Reports

You define a report by assigning values to report tags using the reports editor or a text editor. You can create a new report definition for each report, or you can open and modify a report definition. You can save the report definition with an RPT or HDE format.

You must specify these attributes for every report:

- Report Type
- Description
- Point of View

You can generate these report types in Financial Management:

- Data Explorer reports, which display information from data grids
- 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 run reports from the System Reports module on the Desktop or from the Reporting tab on the Web.

Tip:

To access the Reporting tab on the Web, click the Administration link in the navigation bar.
Caution!

When you define reports, you must follow the syntax as shown in the following examples. The ReportType must start with a capital letter. Do not include spaces between words or on either side of the equal sign.

The following example shows a Report Type, Report Description, and Point of View for a Journal report.

```
ReportType=Journal
ReportDescription=TaxJournals
```

The Point of View definitions are different for each report. For detailed examples of reports, see “Data Explorer Report Example” on page 196, “Journal Report Example” on page 197, and “Creating Intercompany Partner Matching Reports ” on page 198.

The date, time, and user fields are automatically displayed as header information on all reports.

When you create a Journals, Data Explorer or Intercompany Partner Matching Report on the Desktop, the report definition is created automatically and opened in the System Reports module. When you create a report from the Journals module on the Desktop, it opens in the View tab in the Reports module. When you create a report from the Journals module on the Web, you can preview the report or save the definition. When you create a report from the Data Explorer module, it opens in the Design tab. You can use the Design view to modify a report script before you generate the report. You can then set a report description, apply a new style sheet, and save the report locally or remotely.

When you use reports on the Web, the system creates temporary files and log files containing information about these operations. You should set up a file transfer directory other than the Financial Management Web directory to store these temporary files. For information about creating this File Transfer directory, see the Hyperion Financial Management — System 9 Installation Guide.

### Data Explorer Report Example

The following example shows a sample Data Explorer report definition script:

```
ReportType=DataExplorer
ReportDescription=Default Data Explorer Grid
Scenario=Budget
Year=2007
Period=July
View=<Scenario View>
Entity=UnitedStates.Connecticut
Value=USD
Account=Sales
ICP=[ICP None]
Custom1=[None]
Custom2=[None]
Custom3=[None]
Custom4=[None]
Scenarios=Budget
```
Years=2007
Periods=
Views=<Scenario View>
Entities=Regional,UnitedStates.Connecticut
Values=USD
Accounts=GrossMargin,Sales
ICPs=[ICP Top]
Custom1s=[None]
Custom2s=[None]
Custom3s=[None]
Custom4s=[None]
ScenarioMemberList=[Hierarchy]
YearMemberList=[Hierarchy]
PeriodMemberList=[Hierarchy]
ViewMemberList=[Hierarchy]
ValueMemberList=[Hierarchy]
AccountMemberList=[Hierarchy]
ICPMemberList=[Hierarchy]
Custom1MemberList=[Hierarchy]
Custom2MemberList=[Hierarchy]
Custom3MemberList=[Hierarchy]
Custom4MemberList=[Hierarchy]
RowDimensions=Account
ColDimensions=Entity
DisplayMode=Data
MetadataDisplayMode=Labels
Row=-~ExchangeRates
Row=-~NetProfit
Row=-~NetIncome
Row=-~GrossMargin
OBPSettings=0

Creating Journal Reports

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 based on criteria that you select in the Journals module. 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 single 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).

Journal Report Example

The following example shows a sample Journal report definition:

ReportType=Journal
ReportDescription=Tax Journals
ShowDescriptions=0
Creating Intercompany Partner Matching Reports

You create Intercompany Partner (ICP) Matching reports to list the intercompany transactions that are eliminated during consolidation. Intercompany transactions are managed across the Intercompany Partner dimension. The Intercompany dimension represents a container for all intercompany balances that might exist for an account. You can enter intercompany transactions through data grids, Data Load, Journals, data forms, or Retrieve Data. 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.

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.

You can create Intercompany Matching reports by using the user interface or by using a text editor to specify report options in a script. To edit a report, you must modify the report script. 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 also 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 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 the report. If you would like 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 translation “on-the-fly”. For example, if you request to run the report in the EUR Value dimension, and the ICP transactions have not yet been translated to EUR (in which case no stored data is found in the database), the system translates the ICP transactions on-the-fly using the default translation rule set up for the application. The translated data is used only for reporting purposes, and is not stored in the database.

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. 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. 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 create 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**

For an intercompany matching report to display only the difference amount, you can suppress the intercompany details when you create a report. 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 the Custom 1, 2, 3 or 4 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 1, 2, 3, or 4. The system sorts the details based on this option and provides a subtotal for the group.

**Defining Intercompany Partner Matching Report Scripts**

You use the keywords in the following section to define Intercompany Partner Matching report scripts. After you create a script, save it with the RPT file name extension.

Sample Intercompany Partner Matching report scripts are installed with Financial Management and are located in the `Hyperion\FinancialManagement\Sample Apps` directory.

**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:

```
 StyleSheet=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#2007.P#July.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, you should not have `AccountEntity_0` more than once within one report.

**Syntax**

```
AccountEntity_0=A#Sales.C1#GolfBalls.C2#Customer2
AccountPartner_0=A#Purchases.C1#GolfBalls.C2#Customer2
```

The following 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

The following 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

The following 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 state intercompany transactions for Revenue and Expense. In this case, you must use the suppress reversed transactions option to avoid the 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`

For an intercompany matching report to display only the difference amount, you can suppress the intercompany details when you create a report. 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 lets you specify a value to see 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**

DisplayCustomn=Boolean

Replace \( n \) with the number of the custom dimension and replace *Boolean* with Yes or No. The default is Yes. For example:

DisplayCustom1=Yes
DisplayCustom2=Yes
DisplayCustom3=No
DisplayCustom4=No

**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 Custom 1, Custom 2, Custom 3, or Custom 4 dimension. This keyword is optional.

Syntax
GroupByCustom=Custom

Replace Custom with the number of the custom dimension by which to group the Custom dimensions in the report. For example:
GroupByCustom=1

**Creating Intercompany Transaction Reports**
You can create these types of intercompany transaction reports:

- IC Transactions - create a list of transactions.
- IC Matching by Account - create matching reports based on accounts selected.
- IC Matching by Transaction ID - create matching reports based on transaction ID.
- IC Template

For more information on creating intercompany transaction reports, see “IC Transaction Report Syntax” on page 207.

**IC Transaction Report Syntax**
The following keywords are used in creating an Intercompany Transaction report script.
<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReportType</td>
<td>Specify the report type. For example, ReportType=ICTransactions</td>
</tr>
<tr>
<td>ReportLabel</td>
<td>Specify the report name. For example, ReportLabel=Transaction Report</td>
</tr>
<tr>
<td>ReportDescription</td>
<td>Specify a report description. For example, ReportDescription=Intercompany Transaction Detail Report</td>
</tr>
<tr>
<td>ReportSecurityClass</td>
<td>Specify the security class for the report. The default is [Default].</td>
</tr>
<tr>
<td>POV</td>
<td>Specify a valid point of view for the report. For example, POV=S#ActMon.Y#2007.P#January</td>
</tr>
<tr>
<td>ScaleFactor</td>
<td>Specify a scale factor for the report. The scale can be a value from 0 to 9.</td>
</tr>
<tr>
<td>Decimal</td>
<td>Specify the number of decimals to display in the report. The number of decimals can be a value from 0 to 9.</td>
</tr>
<tr>
<td>IncludeMatched</td>
<td>Specify True to include matched transactions in the report, otherwise False.</td>
</tr>
<tr>
<td>IncludeUnMatched</td>
<td>Specify True to include unmatched transactions in the report, otherwise False.</td>
</tr>
<tr>
<td>IncludeMisMatched</td>
<td>Specify True to include mismatched transactions in the report, otherwise False.</td>
</tr>
<tr>
<td>IncludePosted</td>
<td>Specify True to include posted transactions in the report, otherwise False.</td>
</tr>
<tr>
<td>IncludeUnPosted</td>
<td>Specify True to include unposted transactions in the report, otherwise False.</td>
</tr>
<tr>
<td>Entity</td>
<td>The entity must be a valid ICP base entity, not a parent entity.</td>
</tr>
<tr>
<td>Partner</td>
<td>The partner entity must be a valid ICP entity for the account.</td>
</tr>
<tr>
<td>Entity Account</td>
<td>If you are displaying Entity transactions, specify an entity account.</td>
</tr>
<tr>
<td>Partner Account</td>
<td>If you are displaying Partner transactions, specify a partner account.</td>
</tr>
<tr>
<td>TransactionID</td>
<td>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/C3/C4 within the Scenario/Year/Period.</td>
</tr>
<tr>
<td>Transaction Sub ID</td>
<td>Specify a transaction Sub ID.</td>
</tr>
<tr>
<td>TransactionCurrency</td>
<td>This is the currency used for the Invoice transaction. It must be a valid currency defined in the application.</td>
</tr>
<tr>
<td>ReferenceID</td>
<td>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.</td>
</tr>
<tr>
<td>MatchCode</td>
<td>This is optional. The match code must be one of the following prefixes to distinguish the different types of matching processes:</td>
</tr>
<tr>
<td></td>
<td>● A - Auto-matching performed using Accounts</td>
</tr>
<tr>
<td><strong>Keyword</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| I - Auto-matching performed using the Transaction ID | ● I - Auto-matching performed using the Transaction ID  
● R - Auto-matching performed using the Reference ID  
● M - Manual matching performed |
| TransactionAmount               | Specify a range of transaction amounts.                                      |
| 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                        | This is optional. This must be a valid date.                                |
| ToDate                          | This is 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: <Displayed Columns>, <Sort>, <Options>, <Repeat>, <Totals> where <Displayed Columns> is a valid column, <Sort> is Ascending or Descending, <Options> is Label, Description or Both, <Repeat> is Repeat or NoRepeat, and <Totals> is Total or NoTotal.  
For example:  
Entity, Ascending, Label, Repeat, Total |
| DisplayEntityTransactions       | Specify True to display entity transactions in the report, otherwise False.  
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 would like to see the corresponding transactions for Entity B with Partner A, you can select to include Entity transactions and Partner transactions. |
| DisplayPartnerTransactions      | Specify True to display partner transactions in the report, otherwise False. |
| Suppress Details                | Specify True to suppress transaction detail and display only the subtotal row. |
Managing Rules

In This Chapter

Rule Types ........................................................................................................................................... 212
Rule Considerations ............................................................................................................................ 213
Rule Execution During Consolidation ................................................................................................. 220
Default Translation ............................................................................................................................ 221
Financial Management Objects .......................................................................................................... 222
Commonly Used Rules ......................................................................................................................... 223
Creating Rules in the Rules Editor ...................................................................................................... 227
Creating Rules Files ............................................................................................................................ 229
Loading Rules ...................................................................................................................................... 230
Extracting Rules ................................................................................................................................... 231

Following is the information that you need to know about how rules are used in Financial Management. You use 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, or you can use the Financial Management Rules Editor to create rules.
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.

The following table describes the types of rules that you can write.

<table>
<thead>
<tr>
<th>Rule Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calculation</td>
<td>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.</td>
</tr>
</tbody>
</table>
| Translation | 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. 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. Financial Management executes Translation rules in these cases:  
  ● When a user runs a currency translation by right-clicking in the Explore Data module and selecting Translate or Force Translate from the menu that is displayed. This is useful for performing translations as you enter data.  
  ● When a user performs a consolidation and a parent entity’s default currency is different from a child entity’s default currency. |
| Consolidation | 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. |
| Allocation | 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 some other entity. |
| Input | 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. |
| NoInput | 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. These limitations and guidelines apply to NoInput rules:  
  ● Only these Financial Management functions are supported for NoInput rules:  
    ○ NoInput  
    ○ List  
  
  Note: All VBScript methods are supported for NoInput rules.  
  ● 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. |
<table>
<thead>
<tr>
<th>Rule Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule Type</td>
<td>Description</td>
</tr>
<tr>
<td>An If...Then</td>
<td>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 If HS.Entity.Member = &quot;CT&quot;.</td>
</tr>
<tr>
<td>Be careful</td>
<td>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.</td>
</tr>
<tr>
<td>Dynamic</td>
<td>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.</td>
</tr>
<tr>
<td>Calculation</td>
<td>Use these guidelines for writing dynamic calculation rules:</td>
</tr>
<tr>
<td></td>
<td>● 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.</td>
</tr>
<tr>
<td></td>
<td>● Only dynamic accounts are valid on the left side of the equation.</td>
</tr>
<tr>
<td></td>
<td>● Dynamic accounts cannot be used on the right side of the equation.</td>
</tr>
<tr>
<td></td>
<td>● Only Account and View are valid on the left side of the equation.</td>
</tr>
<tr>
<td></td>
<td>● 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.</td>
</tr>
<tr>
<td></td>
<td>● HS.View.PeriodNumber is the only HS statement that can be used in a HS.Dynamic calculation.</td>
</tr>
<tr>
<td></td>
<td>● All statements in the Sub Dynamic section are executed sequentially.</td>
</tr>
<tr>
<td>Transactions</td>
<td>Transactions rules specify the accounts in the application that support intercompany transactions. Cells supporting transactions are read-only in Web grids and data forms.</td>
</tr>
</tbody>
</table>

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 desired 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” on page 214.
- The dimension members to which Financial Management applies calculations depend on the Data Explorer cell where the user’s cursor is placed and the members specified in the Point of View bar. See “Current Dimension Members” on page 217.
- 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” on page 219.
• 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” on page 220.

**Calculation Commands**

Calculate routines are executed for a given 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 fill fail if you have the [None] entity in a hierarchy when you consolidate.

Users run these processes in the Data Explorer 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 a user selects a calculation command, Financial Management executes the Sub Calculate () routine in the RLE file. The Calculate() routine calculates accounts and custom dimension members for a given Entity-Value combination within a given 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> will run rules on the <entity currency> member only. Force Calculate on the <entity currency total> will run rules on <entity currency>, <entity currency adj>, and <entity currency total>. Force Calculate is the only command 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. The following sections explain how this works.
Caution!

Carefully read the following 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” on page 224.

When a user selects the Calculate or Force Calculate command, 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, 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. The following list describes how this works:

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 AllowAdj attribute enabled. The UnitedStates entity has the AllowAdj 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 the following intersections of the Entity and Value dimensions:

2. EastUS and <Entity Currency Adjs>. (EastUS’s AllowAdj 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].
7. EastUS and [Contribution Adjs]. (UnitedStates’ AllowAdjFromChildren attribute has been set to Y.)
9. WestUS and <Entity Currency Adjs>. (WestUS’s AllowAdj 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 is an example of the consolidation process.
Current Dimension Members

By default, 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” on page 214. The current dimension members are determined by the following factors:

- The cell in Data Explorer 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” on page 215.

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 Financial Management applies the function.

The Account Expression characters are listed in the following table.

<table>
<thead>
<tr>
<th>Character</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>A#</td>
<td>Account</td>
</tr>
<tr>
<td>I#</td>
<td>Intercompany Partner</td>
</tr>
<tr>
<td>C1#</td>
<td>Custom1</td>
</tr>
<tr>
<td>C2#</td>
<td>Custom2</td>
</tr>
<tr>
<td>C3#</td>
<td>Custom3</td>
</tr>
<tr>
<td>C4#</td>
<td>Custom4</td>
</tr>
<tr>
<td>S#</td>
<td>Scenario</td>
</tr>
<tr>
<td>Y#</td>
<td>Year</td>
</tr>
<tr>
<td>P#</td>
<td>Period</td>
</tr>
<tr>
<td>W#</td>
<td>View</td>
</tr>
<tr>
<td>E#</td>
<td>Entity</td>
</tr>
</tbody>
</table>
To understand Account Expressions, consider the Clear function, which removes values from the dimension members specified in the function’s argument. In the following 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 name of the member of the Account dimension to which Financial Management applies the Clear function. In other words, 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 the following dimensions:

- Entity
- Period
- Scenario
- Value
- View
- Year
- Custom - Uses the CustomTop member that was set for the account in the metadata. For example, if the Account Expression does not specify a member of the Custom 2 dimension, Financial Management uses all valid Custom 2 members as defined by the CustomTop member 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, the following 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, Financial Management automatically clears data from the member if either of the following 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, and so on. 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, and so on.

**Error Messages**

When Financial Management detects a syntax error, it displays an error message that contains the following information:

- The number of the line in the RLE file that is causing the error.
- The applicable Financial Management object and function.
- A description of the error.

**Rule Execution During Consolidation**

During the consolidation process, rules are executed in a pre-defined sequence. For each base child of a given parent, the calculation sequence for the various elements in the Value dimension takes place in the following 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 attribute Flow or Balance, 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, the following 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 Financial Management objects when using functions.

HS Object

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.

The following 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. The following objects are children of the HS object and are used in Financial Management:

- Account
- AppSettings
- Custom1 through Custom4
- DataUnit
- Entity
Commonly Used Rules

The following 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” on page 229.

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” on page 217). In addition, you can use Account Expression characters to override the current Custom and ICP members.

In the following 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. For more information and examples, see “Exp” on page 266. To quickly illustrate how these characters work, the following example inserts the data from the previous year’s intersection of the PrevCash account and the Golf member of the Custom 1 dimension into the current year’s intersection of PrevCash and Golf:

HS.EXP "A#PrevCash.C1#Golf = A#Cash.Y#Prior.C1#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” on page 267.

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 the following 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 Financial Management determines the current dimension members, see “Current Dimension Members” on page 217.

To do this, use the `Member` function in a VBScript `If` structure. `Member` gets the name of the current member of the following 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:
If structures have many aspects. The following sections cover only a few of these 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 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 the following 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 the following example, if the current scenario is Budget or Forecast, 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 the following example, different accounts will be updated depending upon whether the current scenario is Budget or Actual:

- 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 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.

The following 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 in the Rules Editor

The Rules Editor provides a graphical interface for creating rules. The Rules Editor can be launched from the Windows Desktop of the client computer. For the user, the program runs off of the client workstation and uses disk space and RAM at the client workstation. After you launch the program, you can create new rules or you can open existing rules.

To start the Rules Editor from the Windows Desktop, double-click the Financial Management Rules Editor icon or select it from the Start menu.

Configuring the Rules Editor

You can configure the color and fonts of the different parts of a rules file. This is useful for quickly identifying the parts of the rules file. For example, you can set Financial Management objects and methods to be a different color than Visual Basic objects.

You can change the color and font of the following parts of a rules file:

- Text
- Text Selections
- Numbers
- Operators
- Comments
- Strings
- Statements
- Financial Management Methods
- Visual Basic Objects
- Properties
- Events
- Functions
- Constants
- Financial Management Objects

To configure the Rules Editor:

1. Select Tools > Configure.
2. For Font and Color Settings, select the part of the rules file to configure.
3. Do one or all of these tasks:
   - From Foreground, select a color for the text for this part of the rules file.
Note:
Make sure the **Automatic** check box is not selected.

- From **Background**, select a color for the background for this part of the rules file.

**Note:**
Make sure the **Automatic** check box is not selected.

- Click **Choose Font** and select the font and font size for the text in this part of the rules file. Then click **OK**.

**Note:**
You can click **Reset All** to reset the colors and fonts for all parts of the rules file to the default settings.

4 Repeat steps 2-3 until you have selected all parts of the rules file to configure.

5 Click **OK**.

**Using the Function Wizard**

You can use the Function Wizard in the Rules Editor to select objects and functions to use in rules. When you select the object and function to use, the fields are available for coding. For example, to build an Exp function, choose the object HS, the function Exp and then select the source point of view and the destination point of view by clicking the POV button.

Rules files use the RLE file extension.

**Showing the Point of View**

You use the Point of View selector to create and edit rules using the metadata stored in your Financial Management application. You can select members to use in the rules.

**Scanning Rules**

You can scan rules to verify that the Visual Basic Script syntax and the metadata are valid. The scan also validates the destination for the rule. If it is not a valid destination, an error occurs. After you scan the file, you can load it directly into the application from the Rules Editor.

➢ To scan rules, select **HFM > Scan rule**.

**Loading Rules**

You can a load rules in an application directly from the Rules Editor.
To load rules to an application, select HFM > Load rule.

**Extracting Rules**

You can extract rules from an open application into the Rules Editor.

To extract rules from an application, select HFM > Extract rule.

**Creating Rules Files**

You can create rules in a text editor such as Microsoft 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 Financial Management functions in rules files. Rules files use the RLE file extension.

You can include all types of Financial Management rules in any order in the rules file. The basic syntax to define each routine is displayed in Table 38:

<table>
<thead>
<tr>
<th>Table 38</th>
<th>Rules Routines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub Calculate()</td>
<td></td>
</tr>
<tr>
<td>Type your Calculation rule here.</td>
<td></td>
</tr>
<tr>
<td>End Sub</td>
<td></td>
</tr>
<tr>
<td>Sub Translate()</td>
<td></td>
</tr>
<tr>
<td>Type your Translation rule here.</td>
<td></td>
</tr>
<tr>
<td>End Sub</td>
<td></td>
</tr>
<tr>
<td>Sub Allocate()</td>
<td></td>
</tr>
<tr>
<td>Type your Allocation rule here.</td>
<td></td>
</tr>
<tr>
<td>End Sub</td>
<td></td>
</tr>
<tr>
<td>Sub Input()</td>
<td></td>
</tr>
<tr>
<td>Type your Input rule here.</td>
<td></td>
</tr>
<tr>
<td>End Sub</td>
<td></td>
</tr>
<tr>
<td>Sub NoInput()</td>
<td></td>
</tr>
<tr>
<td>Type your NoInput rule here.</td>
<td></td>
</tr>
<tr>
<td>End Sub</td>
<td></td>
</tr>
<tr>
<td>Sub Consolidate()</td>
<td></td>
</tr>
<tr>
<td>Type your Consolidation rule here.</td>
<td></td>
</tr>
<tr>
<td>End Sub</td>
<td></td>
</tr>
<tr>
<td>Sub Dynamic()</td>
<td></td>
</tr>
<tr>
<td>Type your Dynamic rule here.</td>
<td></td>
</tr>
</tbody>
</table>
End Sub

Sub Transactions()
    'Type your Transactions rule here.
End Sub

Loading Rules

After you load a rules file to an application, anyone using that application is notified that the system has changed and that the user must log off from the application and log back on.

For intercompany transactions, there is an option to check 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.

Windows Procedure

To load rules:
1 Open the application.
2 From the navigation frame, select Load Rules.
3 For Rules Filename, enter the name of the file to load, or click \ to find the file.

Note:
Rules files use the RLE file extension.

4 For Log Filename, enter a log file name, or click \ to find the file.
5 Optional: Select Check Integrity for Intercompany Transactions to verify that posted transactions are valid with the statements in the Sub Transactions section of the rule file that you are loading.
6 Optional: Click Scan to verify that the file format is correct.
7 Click Load.

Caution!
If you make changes to existing rules, you must load the updated RLE file into the application before your changes take effect.

Web Procedure

Note:
Hyperion recommends that you add Financial Management to the exceptions for your Web pop-up blocker. When you perform some Financial Management tasks on the Web 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 Open the application.
2 In the Browser View, expand Tasks and select Load Tasks.
3 Select Load Rules.
4 For Rules File, enter the name of the file to load, or click Browse to find the file.

Note:
Rules files use the RLE file extension.
5 Optional: Select Check Integrity for Intercompany Transactions to verify that posted transactions are valid with the statements in the Sub Transactions section of the rule file that you are loading.
6 Optional: Click Scan to verify that the file format is correct.
7 Click Load.

Extracting Rules

When you extract rules, they are saved to an ASCII file that supports multibyte character sets (MBCS). Rules files use the RLE file extension. After you extract rules, you can view and modify them in a text editor.

Windows Procedure

➤ To extract rules:
1 Open the application.
2 From the navigation frame, select Extract Rules.
3 For Rules Filename, enter a file name to extract, or click Browse to find the file.

Note:
Rules files use the RLE file extension.
4 For Log Filename, enter a log file name, or click Browse to find the file.
5 Click Extract.

Note:
After you extract rules, you can click the View button next to Log Filename to display the log file which notes errors encountered during the extract.
Web Procedure

➢ To extract rules:

1. Open an application.

2. In the Browser View, expand Tasks and select Extract Tasks.


4. 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.
Rule Functions

In This Chapter

Functions Overview ................................................................. 236
ABSExp ........................................................................... 242
AccountType .................................................................... 243
AccountTypeID .................................................................... 244
AddEntityToList ................................................................. 245
AddEntityToListUsingIDs ..................................................... 246
AddMemberToList ............................................................... 247
AddMemberToListUsingIDs ................................................... 247
Alloc ............................................................................. 248
AllowAdjFromChildren ....................................................... 249
AllowAdjs ......................................................................... 250
CalculateExchangeRate ....................................................... 250
CalculateRate ................................................................... 251
CalcStatus ........................................................................ 252
Clear ............................................................................. 253
Con ................................................................................ 254
Consol1, Consol2, Consol3 ................................................ 255
ConsolidateYTD ................................................................. 256
ContainsCellText ............................................................... 256
CreateInputCache .............................................................. 257
Currency .......................................................................... 258
Custom1Top, Custom2Top, Custom3Top, and Custom4Top ................................................................. 258
Decimal ........................................................................... 259
DefaultFreq ..................................................................... 260
DefaultTranslate ................................................................ 261
DefaultView ..................................................................... 261
DefCurrency ..................................................................... 262
DOwn ............................................................................... 263
Dynamic .......................................................................... 264
EnableProcessManagement ............................................... 265
Exp ................................................................................ 266
GetCell ........................................................................... 272
GetCellNoData ................................................................... 273
GetCellRealData .......................................................... 274
GetCellType ................................................................. 275
GetItem ................................................................. 275
GetItemIds2 ................................................................. 276
GetNumItems .......................................................... 277
GetNumLD ................................................................. 277
GetRate ................................................................. 278
GetSubmissionGroup ................................................. 279
GetSubmissionPhase .................................................. 279
Holding ................................................................. 280
ICPTopMember .......................................................... 280
ICPWeight .............................................................. 281
IDFromMember ........................................................ 282
ImpactStatus ............................................................. 283
Input ................................................................. 284
IsAlmostEqual ........................................................... 284
IsBase ................................................................. 285
IsCalculated ............................................................. 286
IsChild ................................................................. 287
IsConsolidated ........................................................ 289
IsDescendant .......................................................... 289
IsFirst ................................................................. 291
IsICP ................................................................. 292
IsLast ................................................................. 293
IsTransCur .............................................................. 293
IsTransCurAdj ........................................................ 294
IsValidDest ............................................................ 294
IsZero ................................................................. 295
List ................................................................. 296
Member .............................................................. 297
MemberFromID ........................................................ 298
Method .............................................................. 299
NoInput ............................................................... 299
NoRound .............................................................. 300
NumBase .............................................................. 301
Number .............................................................. 302
NumChild ............................................................ 303
NumDescendant ........................................................ 304
OpenDataUnit .......................................................... 306
PCon ................................................................. 307
PlugAcct .............................................................. 307
POwn ............................................................... 308
PVAForBalance ........................................................ 309
You can write the following types of rules using these functions:

- Calculation rules
- Translation rules
- Consolidation rules
- Allocation rules
- Input rules
- NoInput rules
- Dynamic calculation rules
- Transactions 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” on page 236.

For more information on types of rules, see Chapter 10, “Managing Rules”.
Table 39 summarizes the 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.

In the following table, rules types are abbreviated as follows:

- Alloc - Allocation
- Calc - Calculation
- Con - Consolidation
- Tran - Translation
- Dyn - Dynamic Calculation
- Trans - Transactions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Objects</th>
<th>Types of Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSExp</td>
<td>Executes a calculation expression and stores the result as an absolute value.</td>
<td>HS</td>
<td>Calc, Tran, Alloc</td>
</tr>
<tr>
<td>AccountType</td>
<td>Gets the account type for the member.</td>
<td>Account</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>AccountTypeID</td>
<td>Gets the account type ID for the member</td>
<td>Account</td>
<td>Calc, Tran, Con, Alloc, Dyn, Trans</td>
</tr>
<tr>
<td>AddEntityToList</td>
<td>Adds the specified entity and parent to a member list.</td>
<td>HS</td>
<td>Calc, Tran, Con, Alloc, Dyn, Trans</td>
</tr>
<tr>
<td>AddEntityToListUsingIDs</td>
<td>Using entity and parent IDs, adds the specified entity and parent to an internal list.</td>
<td>HS</td>
<td>Calc, Tran, Con, Alloc, Dyn, Trans</td>
</tr>
<tr>
<td>AddMemberToList</td>
<td>Adds the member to the member list.</td>
<td>HS</td>
<td>Calc, Tran, Con, Alloc, Dyn, Trans</td>
</tr>
<tr>
<td>AddMemberToListUsingIDs</td>
<td>Using member IDs, adds the specified member to an internal list.</td>
<td>HS</td>
<td>Calc, Tran, Con, Alloc, Dyn, Trans</td>
</tr>
<tr>
<td>Alloc</td>
<td>Allocates data to a cell.</td>
<td>HS</td>
<td>Alloc</td>
</tr>
<tr>
<td>AllowAdjFromChildren</td>
<td>Determines if journal postings from children are allowed for the member.</td>
<td>Entity, Parent</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>Objects</td>
<td>Types of Rules</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>--------------</td>
<td>------------------</td>
</tr>
<tr>
<td>AllowAdjs</td>
<td>Determines if journal postings are allowed for the member.</td>
<td>Entity, Parent</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>CalculateExchangeRate</td>
<td>Calculates the exchange rate from one currency to another.</td>
<td>HS</td>
<td>Calc, Tran, Con, Alloc, Dyn, Trans</td>
</tr>
<tr>
<td>CalculateRate</td>
<td>Gets the current exchange rate for the specified entity.</td>
<td>HS</td>
<td>Calc, Tran, Con, Alloc, Dyn, Trans</td>
</tr>
<tr>
<td>CalcStatus</td>
<td>Gets the calculation status for the cell.</td>
<td>HS</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>Clear</td>
<td>Removes data from a cell.</td>
<td>HS</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>Con</td>
<td>Puts data into the [Proportion] and [Elimination] accounts.</td>
<td>HS</td>
<td>Con</td>
</tr>
<tr>
<td>Consol1, Consol2, Consol3</td>
<td>Gets the Consol1...3 system account value for the node.</td>
<td>Node</td>
<td>Calc, Alloc</td>
</tr>
<tr>
<td>ConsolidateYTD</td>
<td>Determines if the scenario is consolidated using the YTD or periodic method.</td>
<td>Scenario</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>ContainsCellText</td>
<td>Determines if the cell contains cell text.</td>
<td>HS</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>CreateInputCache</td>
<td>Creates a buffer to be used when data is being consolidated.</td>
<td>HS</td>
<td>Calc, Tran, Con, Alloc, Dyn, Trans</td>
</tr>
<tr>
<td>Currency</td>
<td>Gets the currency type for the application or the value member.</td>
<td>AppSettings, Value</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>Custom1Top, Custom2Top, Custom3Top, and Custom4Top</td>
<td>Gets the Top Custom member.</td>
<td>Account</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>Decimal</td>
<td>Gets the number of decimal places for the specified account</td>
<td>Account</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>DefaultFreq</td>
<td>Gets the default scenario frequency.</td>
<td>Scenario</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>DefaultTranslate</td>
<td>Calculates translation by bypassing rules.</td>
<td>HS</td>
<td>Trans</td>
</tr>
<tr>
<td>DefaultView</td>
<td>Gets the default scenario view.</td>
<td>Scenario</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>Objects</td>
<td>Types of Rules</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>--------------</td>
<td>------------------</td>
</tr>
<tr>
<td>DefCurrency</td>
<td>Gets the default currency for the entity or parent.</td>
<td>Entity, Parent</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>DOwn</td>
<td>Gets the percentage of Direct Ownership (DOwn) for the node.</td>
<td>Node</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>Dynamic</td>
<td>Specifies the formula for the dynamic accounts that need calculations.</td>
<td>HS</td>
<td>Dyn</td>
</tr>
<tr>
<td>EnableProcessManagement</td>
<td>Determines if a scenario supports process management.</td>
<td>Scenario</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>Exp</td>
<td>Executes a calculation expression and puts data into a specified point of view.</td>
<td>HS</td>
<td>Calc, Tran, Alloc</td>
</tr>
<tr>
<td>GetCell</td>
<td>Gets the data contained in a cell.</td>
<td>HS</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>GetCellNoData</td>
<td>Gets the data contained in a cell and indicates if the cell contains no data.</td>
<td>HS</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>GetCellRealData</td>
<td>Gets the data contained in a cell and indicates if the cell contains real data.</td>
<td>HS</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>GetCellType</td>
<td>Gets the cell type.</td>
<td>HS</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>GetItem</td>
<td>Gets an individual record to process for consolidation.</td>
<td>DataUnit</td>
<td>Con</td>
</tr>
<tr>
<td>GetItemIDs2</td>
<td>Gets an individual record to process for consolidation using dimension ID numbers.</td>
<td>DataUnit</td>
<td>Con</td>
</tr>
<tr>
<td>GetNumItems</td>
<td>Gets the number of records to process for consolidation.</td>
<td>DataUnit</td>
<td>Con</td>
</tr>
<tr>
<td>GetNumLID</td>
<td>Gets the number of line-item details for the specified POV</td>
<td>HS</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>GetRate</td>
<td>Gets the currency rate for a cell.</td>
<td>HS</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>Objects</td>
<td>Types of Rules</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>GetSubmissionGroup</td>
<td>Gets the submission group for a cell.</td>
<td>HS</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>GetSubmissionPhase</td>
<td>Gets the submission phase for a cell.</td>
<td>HS</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>Holding</td>
<td>Gets the holding company for the member.</td>
<td>Entity, Parent</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>ICPTopMember</td>
<td>Gets the ICP Top Member of the current Account dimension member or the specified account member.</td>
<td>Account</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>ICPWeight</td>
<td>Gets the percentage of ICP entity balances that aggregate to the [ICP Top] value member.</td>
<td>AppSettings</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>IDFromMember</td>
<td>Gets the ID number for the specified member.</td>
<td>Account, Custom1 through Custom4, Entity, ICP, Parent, Year, Period, Scenario, Value, View</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>ImpactStatus</td>
<td>Changes the status of the specified data unit to impacted.</td>
<td>HS</td>
<td>Calc</td>
</tr>
<tr>
<td>Input</td>
<td>Enables input at parent entity level for specified account.</td>
<td>HS</td>
<td>Input</td>
</tr>
<tr>
<td>IsAlmostEqual</td>
<td>Checks to see if two values are equal</td>
<td>HS</td>
<td>Calc, Tran, Con, Alloc, Dyn, Trans</td>
</tr>
<tr>
<td>IsBase</td>
<td>Determines if the member is a base member.</td>
<td>Account, Custom1 through Custom4, Entity, Parent, Node</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>IsCalculated</td>
<td>Determines if the account is calculated.</td>
<td>Account</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>IsChild</td>
<td>Determines if the member is a child of another member.</td>
<td>Account, Custom1 through Custom4, Entity, Parent, Node</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>IsConsolidated</td>
<td>Determines if the account is consolidated.</td>
<td>Account</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>IsDescendant</td>
<td>Determines if the member is a descendant of another member.</td>
<td>Account, Custom1 through Custom4, Entity, Parent, Node</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>Objects</td>
<td>Types of Rules</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------------------------------------------------------</td>
<td>------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>IsFirst</td>
<td>Determines if the period or year is the first for the application.</td>
<td>Period, Year</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>IsICP</td>
<td>Determines if the member is an ICP.</td>
<td>Account, Entity, Parent</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>IsLast</td>
<td>Determines if the year or period is the last for the application.</td>
<td>Period, Year</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>IsTransCur</td>
<td>Determines if the value member is a translated currency member.</td>
<td>Value</td>
<td>Tran</td>
</tr>
<tr>
<td>IsTransCurAdj</td>
<td>Determines if the value member is a translated currency Adj member.</td>
<td>Value</td>
<td>Tran</td>
</tr>
<tr>
<td>IsValidDest</td>
<td>Determines if the specified POV is a valid destination.</td>
<td>HS</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>IsZero</td>
<td>Checks to see if the passed in value is zero.</td>
<td>HS</td>
<td>Calc, Tran, Con, Alloc, Dyn, Trans</td>
</tr>
<tr>
<td>List</td>
<td>Gets the elements contained in a list.</td>
<td>Account, Custom1 through Custom4, Entity, Parent, ICP, Node, Scenario</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>Member</td>
<td>Gets the name of the member.</td>
<td>Entity, Parent, Period, Scenario, Value, Year, View</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>MemberFromID</td>
<td>Gets the member for the specified ID number.</td>
<td>Account, Custom1 through Custom4, Entity, ICP, Parent, Year, Period, Scenario, Value, View</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>Method</td>
<td>Gets the consolidation method for the member.</td>
<td>Node</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>NoInput</td>
<td>Prevents input into cells.</td>
<td>HS</td>
<td>Noinput</td>
</tr>
<tr>
<td>NoRound</td>
<td>Turns off rounding.</td>
<td>HS</td>
<td>Calc, Tran, Alloc</td>
</tr>
<tr>
<td>NumBase</td>
<td>Gets the number of base members.</td>
<td>Account, Custom1 through Custom4, Entity, Parent, Node, Period</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>Number</td>
<td>Gets the number of the current period.</td>
<td>Period</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>Objects</td>
<td>Types of Rules</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>NumChild</td>
<td>Gets the number of children for the member.</td>
<td>Account, Custom1 through Custom4, Entity, Parent, Node</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>NumDescendant</td>
<td>Gets the number of descendants for the member.</td>
<td>Account, Custom1 through Custom4, Entity, Parent, Node</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>OpenDataUnit</td>
<td>Gets a data unit for consolidation.</td>
<td>HS</td>
<td>Calc, Tran, Con</td>
</tr>
<tr>
<td>PCon</td>
<td>Gets the percentage of consolidation.</td>
<td>Node</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>PlugAcct</td>
<td>Gets the plug account.</td>
<td>Account</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>POwn</td>
<td>Gets the percentage of ownership.</td>
<td>Node</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>PVAForBalance</td>
<td>Determines default translation method for BALANCE accounts.</td>
<td>AppSettings</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>PVAForFlow</td>
<td>Determines default translation method for FLOW accounts.</td>
<td>AppSettings</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>RateForBalance</td>
<td>Gets the default rate for balance.</td>
<td>AppSettings</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>RateForFlow</td>
<td>Gets the default rate for flow.</td>
<td>AppSettings</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>ReviewStatus</td>
<td>Gets the process management review status for the cell.</td>
<td>HS</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>ReviewStatusUsingPhaseID</td>
<td>Gets the process management review status by phase ID for the cell.</td>
<td>HS</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>Round</td>
<td>Rounds the data.</td>
<td>HS</td>
<td>Calc, Tran, Con</td>
</tr>
<tr>
<td>Scale</td>
<td>Gets the scale of the specified currency.</td>
<td>Currency</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>SecurityAsPartner</td>
<td>Gets the security class for the ICP entity.</td>
<td>Entity, Parent</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>SecurityClass</td>
<td>Gets the security class for a dimension member.</td>
<td>Account, Scenario, Entity, Parent, Custom1 through Custom4</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>Objects</td>
<td>Types of Rules</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>SetData</td>
<td>Sets an individual record.</td>
<td>HS</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>SetDataWithPOV</td>
<td>Inserts data into the node or currency cube.</td>
<td>HS</td>
<td>Calc, Tran, Con, Alloc, Dyn, Trans</td>
</tr>
<tr>
<td>SubmissionGroup</td>
<td>Gets the process management submission group for a dimension member.</td>
<td>HS</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>SupportsTran</td>
<td>Defines the accounts in the application that require Intercompany Transaction detail support.</td>
<td>Scenario, Year, Entity, Account, C1...4</td>
<td>Trans</td>
</tr>
<tr>
<td>SwitchSign</td>
<td>Determines if credits are switched to debits for the member.</td>
<td>Custom1 through Custom4</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>SwitchType</td>
<td>Determines if account types are switched for the member.</td>
<td>Custom1 through Custom4</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>Trans</td>
<td>Translates using YTD method.</td>
<td>HS</td>
<td>Tran</td>
</tr>
<tr>
<td>TransPeriodic</td>
<td>Translates using periodic method.</td>
<td>HS</td>
<td>Tran</td>
</tr>
<tr>
<td>UD1...3</td>
<td>Gets the user-defined attribute for the member.</td>
<td>Account, Entity, Parent, Scenario, Custom1 through Custom4</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>ValidationAccount</td>
<td>Gets the validation account.</td>
<td>AppSettings</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>ValidationAccountEx</td>
<td>Gets the validation account for the process management submission phase.</td>
<td>AppSettings</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
<tr>
<td>XBRLTags</td>
<td>Gets the XBRL tag for the account.</td>
<td>Account</td>
<td>Calc, Tran, Con, Alloc</td>
</tr>
</tbody>
</table>

**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 the following types of rules:

- Calculation
- Translation
- Allocation

Syntax

\[ \text{HS.ABSExp } "\text{DestPOV} = \text{Expression}" \]

Table 40  Syntax for ABSExp Function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textit{DestPOV}</td>
<td>A destination point of view that identifies where to put the data&lt;br&gt;You must specify an Account member, and you can optionally specify ICP and Custom members. Note the following usage rules:&lt;br&gt;● If you do not specify an ICP member, the default is [ICP None].&lt;br&gt;● 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 Financial Management validation checks to avoid writing data to invalid intersections.</td>
</tr>
<tr>
<td>\textit{Expression}</td>
<td>A calculation expression</td>
</tr>
</tbody>
</table>

Return Value

None.

Example

The following example sets the amount in the StateTax account. The example calculates the absolute amount by multiplying the amount in the Sales account for 1999 by the rate in the StateRate account for 1999.

\[ \text{HS.ABSExp } "\text{A#StateTax} = \text{A#Sales.Y#1999} \times \text{A#StateRate.Y#1999}" \]

\textbf{AccountType}

Gets the account type for the current Account member or for a specified account. This function can be used in the following types of rules:
- Calculation
- Translation
- Consolidation
- Allocation

Syntax

\[ \text{HS.Account.AccountType}("\text{Account}") \]

\[ \text{HS.Account.AccountType}(" ") \]

where, \textit{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 routine. 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
- FLOW
- BALANCE
- BALANCERECURRING
- CURRENCYRATE
- GROUPLABEL

Example

In the following example if the account type for the Sales account is REVENUE, then statements between the If...Then and End If statements are executed.

```csharp
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 the following 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 routine. Otherwise, you must specify an account when using this function.

Return Value

The ID for the specified account. Table 41 lists valid account types with corresponding IDs.

<table>
<thead>
<tr>
<th>Account Type</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>REVENUE (INCOME in previous releases)</td>
<td>0</td>
</tr>
<tr>
<td>EXPENSE</td>
<td>1</td>
</tr>
<tr>
<td>ASSET</td>
<td>2</td>
</tr>
<tr>
<td>LIABILITY</td>
<td>3</td>
</tr>
<tr>
<td>BALANCE</td>
<td>4</td>
</tr>
<tr>
<td>FLOW</td>
<td>5</td>
</tr>
<tr>
<td>CURRENCYRATE</td>
<td>7</td>
</tr>
<tr>
<td>GROUPLABEL</td>
<td>10</td>
</tr>
<tr>
<td>BALANCERECURRING</td>
<td>11</td>
</tr>
<tr>
<td>DYNAMIC</td>
<td>12</td>
</tr>
</tbody>
</table>

Example

If HS.Account.AccountTypeID("Investments") = 2 Then
... End If

AddEntityToList

Adds the specified entity and parent to a member list. This function can be used in the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation
- Dynamic Calculation
- Transactions

Syntax
HS.AddEntityToList("Parent", "Child")

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent</td>
<td>Name of a valid parent entity.</td>
</tr>
<tr>
<td>Child</td>
<td>Name of a child of the parent entity.</td>
</tr>
</tbody>
</table>

Return Value
None

Example
HS.AddEntityToList "UnitedStates", "Maine"

AddEntityToListUsingIDs

Using the ID for the entity and parent, adds the specified entity and parent to the internal Financial Management list to be used later. This function can be used in the following types of rules:
- Calculation
- Translation
- Consolidation
- Allocation
- Dynamic Calculation
- Transactions

Syntax
HS.AddEntityToListUsingIDs(ParentID, ChildID)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ParentID</td>
<td>The ID for the parent entity.</td>
</tr>
<tr>
<td>ChildID</td>
<td>The ID for the child of the parent entity.</td>
</tr>
</tbody>
</table>
AddMemberToList

Adds the specified member to a member list. This function can be used in the following types of rules:
- Calculation
- Translation
- Consolidation
- Allocation
- Dynamic Calculation
- Transactions

Syntax
HS.AddMemberToList("Member")

where Member is the name of a valid dimension member.

Example
HS.AddMemberToList "July"

AddMemberToListUsingIDs

Adds the specified member to the internal Financial Management list to be used later. This function can be used in the following types of rules:
- Calculation
- Translation
- Consolidation
- Allocation
- Dynamic Calculation
- Transactions

Syntax
HS.AddMemberToListUsingIDs(MemberID)

where MemberID is the ID for a valid dimension member.
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 44  Syntax for Alloc Function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| SourcePOV | A source point of view for the data to be allocated. You must specify an Account member, and you can optionally specify ICP and Custom members. If you do not specify ICP and Custom members:  
  - The default ICP member is [ICP Top].  
  - The default Custom member is <Top Member> for that account. |
| DestPOV   | A destination point of view that identifies where to allocate the data. You must specify an Account member, and you can optionally specify ICP and Custom members. Note the following usage rules:  
  - If you do not specify an ICP member, the default is [ICP None].  
  - If you do not specify Custom members, the default is [None].  
  - If you specify an Entity member, it is used as the parent if the EntityList argument is [Base]. |
| EntityList| A member list that identifies the entities to which the data is be allocated. You can use the system-defined entity list [Base] or you can use a user-defined list. If you use the system-defined list [Base], 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. |
| AllocExp  | An expression that identifies the data to be allocated to each entity. This expression can contain the following types of values:  
  - Numbers  
  - 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 the following usage rules:  
    - If you do not specify Custom members, the default is <Top Member>.  
    - If you do not specify an ICP member, the default is [ICP Top].  
    - If you do not specify Scenario, Year, Period, View, or Value members, the default is <Current>.  
    - If you do not specify an Entity member, the default is the destination entity. |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| PlugAccount | 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. You must specify an Account member, and you can optionally specify ICP and Custom members. Note the following rules of usage:  
- 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>.  
- If you do not specify an ICP member, the default is [ICP None].  
- If you do not specify Custom members, the default is [None]. |

**Return Value**

None.

**Example**

In the following example, data from the TangibleAssets account is allocated to the Cash account. Sum adds up the data for the intersecting cells of the NewEngland member list entities and the TangibleAssets account.

Call HS.ALLOC

("A#TangibleAssets","A#Cash","NewEngland","(A#TangibleAssets / 
HS.Entity.Sum(A#TangibleAssets))","A#Plug")

---

**AllowAdjFromChildren**

Specifies if journal postings from children are allowed for the specified entity. This function can be used in the following 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(" ")
```

where *Entity* is the name of a valid Entity member.

Use a blank string (" ") to apply this function to the current entity.
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 the following example, if journal postings from children of France are allowed, then statements between the If…Then and End If statements are executed.

```vbnet
If HS.Entity.AllowAdjFromChildren("France") = TRUE then
  ...
End If
```

**AllowAdjs**

Specifies if journal postings are allowed for the specified entity. This function can be used in the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**

```vbnet
HS.Entity.AllowAdjs("Entity")
HS.Entity.AllowAdjs(" ")
```

where `Entity` is the name of a valid Entity member.

Use a blank string (" ") to apply this function to the current entity.

**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 the following example, if journal postings for France are allowed, then statements between the If…Then and End If statements are executed.

```vbnet
If HS.Entity.AllowAdjs("France") = TRUE then
  ...
End If
```

**CalculateExchangeRate**

Calculate the exchange rate from one currency to another.
Syntax

`HS.CalculateExchangeRate (ScenarioID, YearID, PeriodID, EntityID, RateAccountID, FromCurrencyID, ToCurrencyID)`

**Table 45  Syntax for CalculateExchangeRate Function**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ScenarioID</td>
<td>The scenario ID.</td>
</tr>
<tr>
<td>YearID</td>
<td>The year ID.</td>
</tr>
<tr>
<td>PeriodID</td>
<td>The period ID.</td>
</tr>
<tr>
<td>EntityID</td>
<td>The entity ID.</td>
</tr>
<tr>
<td>RateAccountID</td>
<td>The rate account ID.</td>
</tr>
<tr>
<td>FromCurrencyID</td>
<td>The ID for the currency that you are going from.</td>
</tr>
<tr>
<td>ToCurrencyID</td>
<td>The ID for the currency that you are going to.</td>
</tr>
</tbody>
</table>

**Return Value**

The exchange rate.

**CalculateRate**

Gets the current exchange rate for the specified entity.

Syntax

`HS.CalculateRate (ScenarioID, YearID, PeriodID, EntityID, CurrID, RateAccountID)`

**Table 46  Syntax for CalculateRate Function**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ScenarioID</td>
<td>The scenario ID.</td>
</tr>
<tr>
<td>YearID</td>
<td>The year ID.</td>
</tr>
<tr>
<td>PeriodID</td>
<td>The period ID.</td>
</tr>
<tr>
<td>EntityID</td>
<td>The entity ID.</td>
</tr>
<tr>
<td>CurrID</td>
<td>The currency ID.</td>
</tr>
<tr>
<td>RateAccountID</td>
<td>The rate account ID.</td>
</tr>
</tbody>
</table>

**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 the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

Syntax

```
HS.CalcStatus("Scenario.Year.Period.Entity.Value")
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario</td>
<td>Name of a valid Scenario member.</td>
</tr>
<tr>
<td>Year</td>
<td>A valid year.</td>
</tr>
<tr>
<td>Period</td>
<td>A valid period.</td>
</tr>
<tr>
<td>Entity</td>
<td>Name of a valid Entity member.</td>
</tr>
<tr>
<td>Value</td>
<td>Name of a valid Value member.</td>
</tr>
</tbody>
</table>

Return Value

A String that contains the calculation status for the specified point of view. Valid status codes are listed below.

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td>None of the data for the specified Scenario, Year, Period, Entity, and Value dimensions has changed.</td>
</tr>
<tr>
<td>OK ND</td>
<td>OK - No data. Some data may have changed.</td>
</tr>
<tr>
<td>OK SC</td>
<td>OK - System changed. Some of the data for the specified Scenario, Year, Period, Entity, and Value dimensions has changed. For example, a new rules file or metadata file has been loaded.</td>
</tr>
<tr>
<td>CH</td>
<td>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 Scenario, Year, Period, Entity, and Value dimension may not be up to date because the SubCalculate routine has not been executed. For base level entities, the data cell may have been entered by manual data entry or by a bulk data file load. For entities, the data cell may have been entered by a journal posting.</td>
</tr>
<tr>
<td>CH ND</td>
<td>Needs Calculation – No Data. This indicates the first time that calculation will be performed on the cell.</td>
</tr>
<tr>
<td>Status Code</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>TR</td>
<td>Needs Translation. The selected Value dimension member is not the entity’s default currency, and its translated values may not be up to date.</td>
</tr>
<tr>
<td>TR ND</td>
<td>Needs Translation — No Data. This indicates the first time that translation will be performed on the cell.</td>
</tr>
<tr>
<td>CN</td>
<td>Needs Consolidation. The data for the specified Scenario, Year, Period, Entity, and Value dimensions may not be up to date because data for a child entity has changed, data for the same entity’s default currency has changed, or metadata parameters or rules have changed.</td>
</tr>
<tr>
<td>CN ND</td>
<td>Needs Consolidation - No Data. The parent has no data but a child of the parent has changed. This indicates the first time that consolidation will be performed on the cell.</td>
</tr>
</tbody>
</table>
| Locked      | The data for the specified Scenario, Year, Period, Entity, and Value dimensions has been locked by an administrator and can no longer be modified by manual data entry or calculations.  
**Note:** You can use the Alloc function to modify data in a locked destination POV. |
| NoData      | No data exists for the specified Scenario, Year, Period, Entity, and Value dimensions. |

**Example**

In the following example, if the status for the specified point of view is “OK”, then statements between the If...Then and End If statements are executed.

```vbnet
    ...
End If
```

**Clear**

Removes data from combinations of Account, ICP, and Custom members. This function can be used in the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**


**Table 49 Syntax for Clear Function**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account</td>
<td>Name of a valid Account member.</td>
</tr>
<tr>
<td>ICP</td>
<td>Name of a valid ICP member.</td>
</tr>
</tbody>
</table>
**Parameter**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom1, Custom2, Custom3, Custom4</td>
</tr>
<tr>
<td>Name of valid Custom1, Custom2, Custom3, and Custom4 members.</td>
</tr>
</tbody>
</table>

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 the following 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. The following 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**

The following 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>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DestPOV</td>
<td>Combination of the following dimensions:</td>
</tr>
<tr>
<td></td>
<td>● Account</td>
</tr>
<tr>
<td></td>
<td>● Custom1, Custom2, Custom3, Custom4</td>
</tr>
<tr>
<td></td>
<td>● Intercompany</td>
</tr>
<tr>
<td></td>
<td>● Entity</td>
</tr>
<tr>
<td></td>
<td>● Value</td>
</tr>
<tr>
<td>Factor</td>
<td>A number or an expression using mathematical operators (+ - * /) or functions like HS.GetCell.</td>
</tr>
<tr>
<td>Nature</td>
<td>A string used for audit purposes. This string is stored and provides information about the accounting purpose of the transaction.</td>
</tr>
</tbody>
</table>
Parameter | Description
--- | ---
 | 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.

Return Value
None.

Example
Call HS.Con ("V#[Elimination]", -1*dPCon, ")

Consol1, Consol2, Consol3

Gets the value in Consol1, Consol2, or Consol3 account for the specified parent.entity node. This function can be used in the following 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 51 Syntax for Consol1, Consol2, Consol3 Functions

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario</td>
<td>Name of a valid Scenario member.</td>
</tr>
<tr>
<td>Year</td>
<td>A valid year.</td>
</tr>
<tr>
<td>Period</td>
<td>A valid period.</td>
</tr>
<tr>
<td>Parent.Entity</td>
<td>Name of a valid Parent.Entity node.</td>
</tr>
</tbody>
</table>

Return Value
The value in the Consol1, Consol2, or Consol3 system account.

Example
The following example gets the Consol1 value for the Group1.Ent1 node in the actual scenario.
ConsolidateYTD

Determines if the current Scenario member or a specified scenario member is consolidated year to date. This function can be used in the following 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 the following 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 the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

Syntax

```
HS.ContainsCellText("POVExpression")
```
where \textit{POVExpression} is a combination of members. If you do not specify a dimension, the following 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

\textbf{Return Value}

A Boolean expression that is True if the specified data cell contains cell text; False otherwise.

\textbf{Example}

In the following example, if the specified cell does not contain cell text, then statements between the \texttt{If...Then} and \texttt{End If} lines are executed.

\begin{verbatim}
If HS.ContainsCellText("A#Sales.C1#Prod1.C2#Region1") = "False" then
... 
End If
\end{verbatim}

\textbf{CreateInputCache}

Creates a buffer to be used when data is being consolidated. This function can be used in the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation
- Dynamic Calculation
- Transactions

\textbf{Syntax}

\begin{verbatim}
HS.CreateInputCache()
\end{verbatim}

\textbf{Return Value}

The cached object.
**Currency**

Gets the currency for the application or for the current Value dimension member. This function can be used in the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**

```
HS.AppSettings.Currency
HS.Value.Currency
```

**Return Value**

A String that contains the name of the currency for the application or for the Value member.

**Example**

In the following 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
```

**Custom1Top, Custom2Top, Custom3Top, and Custom4Top**

Gets the C1Top through C4Top member for the current Account member or for a specified account member. This function can be used in the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**

```
HS.Account.<Object>("Account")
HS.Account.<Object>("")
```

**Note:**

Use a blank string (" ") to apply this function to the current member.
Table 52 Syntax for C1Top...C4Top Function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Object&gt;</td>
<td>One of the following object keywords:</td>
</tr>
<tr>
<td></td>
<td>● C1Top</td>
</tr>
<tr>
<td></td>
<td>● C2Top</td>
</tr>
<tr>
<td></td>
<td>● C3Top</td>
</tr>
<tr>
<td></td>
<td>● C4Top</td>
</tr>
<tr>
<td>Account</td>
<td>Name of a valid Account member.</td>
</tr>
</tbody>
</table>

Return Value

The top member for the specified Custom account member.

Example

In the following example, the C1Top member is the TotalCosts account.

```java
strVariable = HS.Account.C1Top("TotalCosts")
```

**Decimal**

Gets the number of decimal places for the specified account. This function can be used in the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**

```java
HS.Account.Decimal("AccountName")
HS.Account.Decimal(""")
```

**Note:**

Use a blank string (" ") to apply this function to the current account when using a Sub Consolidate routine.

```java
HS.Account.Decimal(Var1)
```

Table 53 Syntax for Decimal Function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccountName</td>
<td>Name of a valid Account member.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>Var1</td>
<td>VBScript variable representing an Account member.</td>
</tr>
</tbody>
</table>

**Return Value**

An integer representing the decimal value assigned to the account. Valid values are 0 to 9.

**Example**

In the following 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.

```vbscript
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 the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**

```vbscript
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 the following example, if the default frequency for the Actual scenario is YTD, then statements between the If...Then and End If statements are executed.

```vbscript
If HS.Scenario.DefaultFreq("Actual") = "YTD" Then
    ...
End If
```
DefaultTranslate

Calculates translation by bypassing rules. This function can be used in Translation rules only.

Syntax

HS.DefaultTranslate(dRateForBalanceAccounts, dRateForFlowAccounts, bUsePVAForFlowAccounts, bUsePVAForBalanceAccounts)

Table 54 Syntax for DefaultTranslate Function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dRateForBalanceAccounts</td>
<td></td>
</tr>
<tr>
<td>dRateForFlowAccounts</td>
<td></td>
</tr>
<tr>
<td>bUserPVAForFlowAccounts</td>
<td></td>
</tr>
<tr>
<td>bUsePVAForBalanceAccounts</td>
<td></td>
</tr>
</tbody>
</table>

Return Value

None.

DefaultView

Gets the default view for the current Scenario member or for a specified scenario member. This function can be used in the following 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 the following example, if the default view for the Actual scenario is YTD, then statements between the If...Then and End If statements are executed.

```vbnet
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. This function can be used in the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

Syntax

```vbnet
HS.<Object>.DefCurrency("Entity")
HS.<Object>.DefCurrency(""
```

Note:

Use a blank string ("" ) to apply this function to the current member.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Object&gt;</td>
<td>One of the following object keywords:</td>
</tr>
<tr>
<td></td>
<td>● Entity</td>
</tr>
<tr>
<td></td>
<td>● Parent</td>
</tr>
<tr>
<td>Entity</td>
<td>Name of a valid Entity member.</td>
</tr>
</tbody>
</table>

Return Value

A string that contains the default currency for the specified entity or parent.
Example

In the following 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 the following 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>
<thead>
<tr>
<th>Table 56</th>
<th>Syntax for DOwn Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>Scenario</td>
<td>Name of a valid Scenario member.</td>
</tr>
<tr>
<td>Year</td>
<td>A valid year.</td>
</tr>
<tr>
<td>Period</td>
<td>A valid period.</td>
</tr>
<tr>
<td>Parent.Entity</td>
<td>Name of a valid Parent.Entity node.</td>
</tr>
</tbody>
</table>

**Return Value**

A number that is a percentage of direct ownership.

**Example**

dVar1 = HS.Node.DOwn("S#Actual.Y#2007.P#Q1.E#Group1.Ent1")
Dynamic

Specifies the formula for the dynamic accounts that need calculations. This function can be used in Dynamic rules.

Note:

You can embed the HS.View.PeriodNumber function in the Dynamic function.

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 a “parking” account to store information from other cubes. For example, to reference a prior year’s data in the formula, you need to use a “parking” account to store the last year’s data information so that it can be referenced in the dynamic calculation within the same cube.

Syntax

HS.Dynamic "DestPOV = Expression"

Table 57 Syntax for Dynamic Function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DestPOV</td>
<td>A valid Account member with type set to Dynamic. You may also specify a view for which to execute the calculation.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> 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.</td>
</tr>
<tr>
<td>Expression</td>
<td>A calculation expression.</td>
</tr>
</tbody>
</table>

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

The following table lists the expected results for the GM% account.

<table>
<thead>
<tr>
<th>Custom1</th>
<th>Sales</th>
<th>GM</th>
<th>GM% (Dynamic Calculation)</th>
<th>Calculation based on formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>600</td>
<td>140</td>
<td>23.33%</td>
<td>140 / 160 * 100</td>
</tr>
<tr>
<td>Custom1</td>
<td>Sales</td>
<td>GM</td>
<td>Custom1% (Dynamic Calculation)</td>
<td>Calculation based on formula</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>-----</td>
<td>-------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>P1</td>
<td>100</td>
<td>10</td>
<td>10%</td>
<td>10 / 100 * 100</td>
</tr>
<tr>
<td>P2</td>
<td>200</td>
<td>40</td>
<td>20%</td>
<td>40 / 200 * 100</td>
</tr>
<tr>
<td>P3</td>
<td>300</td>
<td>90</td>
<td>30%</td>
<td>90 / 300 * 100</td>
</tr>
</tbody>
</table>

### EnableProcessManagement

Determines if a scenario supports process management. This function can be used in the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

### Syntax

```vbnet
HS.Scenario.EnableProcessManagement("Scenario")
HS.Scenario.EnableProcessManagement(""")
```

**Note:**

Use a blank string ("") to apply this function to the current scenario.

```
HS.Scenario.EnableProcessManagement(Var1)
```

### Table 58  Syntax for EnableProcessManagement Function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario</td>
<td>A valid scenario.</td>
</tr>
<tr>
<td>Var1</td>
<td>VBScript variable representing a Scenario member.</td>
</tr>
</tbody>
</table>

### Return Value

A Boolean that is True if the scenario has process management enabled; False otherwise.

### Example

In the following example, if process management is enabled for the actual scenario, statements between the If and End If statements are executed:

```vbnet
If HS.Scenario.EnableProcessManagement("Actual") = "TRUE" then
    ...
End IF
```
Exp

Puts data into a combination of Account, ICP, and Custom1...4 members. This function can be used in the following types of rules:

- Calculation
- Translation
- Allocation

Syntax

HS.Exp "DestPOV = Expression"

Table 59  Syntax for Exp Function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| DestPOV   | A destination point of view that identifies where to put the data. You must specify an Account member, and you can optionally specify ICP and Custom members. Note the following usage rules:  
  - If you do not specify an ICP member, the default is [ICP None].  
  - 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 Financial Management validation checks to avoid writing data to invalid intersections. |
| Expression | 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. The following 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, not just the Intercompany Partner and Custom dimensions. Thus, you can assign data for a group of cells to another group of cells. The following 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” on page 268.
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 the following 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, Financial Management will insert data into all accounts that are valid for the current point of view. See “Simultaneously Setting Several Accounts” on page 268.

- **Source.** If a member of a dimension is not specified on the right side of the equal sign, then 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” on page 269.

Period and Year Keywords

To create dynamic rules, you can use the keywords in Table 60 instead of member names to represent members of the destination Year or Period dimensions:

<table>
<thead>
<tr>
<th><strong>Table 60</strong></th>
<th>Period and Year Keywords and Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Keyword</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>CUR</td>
<td>The current period or year.</td>
</tr>
<tr>
<td>FIRST</td>
<td>The first period or year that is valid for the application.</td>
</tr>
<tr>
<td>LAST</td>
<td>The last period or year that is valid for the application.</td>
</tr>
<tr>
<td>Keyword</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>NEXT</td>
<td>The period or year that follows the current period or year.</td>
</tr>
<tr>
<td>PRIOR</td>
<td>The period or year that precedes the current period or year.</td>
</tr>
</tbody>
</table>

**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. The following example sets the MiscPast account to the amount in the Misc account two years before the current year.

\[
\text{HS.Exp} \ "A\#MiscPast = A\#Misc.Y\#CUR-2"
\]

**Mathematical Calculations**
You can add, subtract, multiply, and divide on the right side of the equal sign. You must use the following standard VBScript characters:

\[ + - * / \]

The following example adds the values in the prior year’s Taxes and Purchases accounts and then inserts the sum in the Cash account.

\[
\text{HS.Exp} \ "A\#Cash = A\#Taxes.Y\#PRIOR + A\#Purchases.Y\#PRIOR"
\]

**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 the following example, the NumBase function is nested in the Exp function, and so its String argument is not surrounded by quotation marks.

\[
\text{HS.Exp} \ "A\#SalesAlloc = A\#Sales/HS.Entity.NumBase(Regional)"
\]

**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 the following 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.P#PRIOR"
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. The following 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
The following example sets the amount in the StateTax account. The example calculates this amount by multiplying the amount in the Sales account for 1999 by the rate in the StateRate account for 1999.

HS.Exp "A#StateTax = A#Sales.Y#1999 * A#StateRate.Y#1999"

Exp and Dimension Intersection Considerations

The following pages contain examples that illustrate the considerations introduced in “Dimension Intersection Considerations” on page 267. The following types of intersections are covered:

- All intersections are valid for the source and destination accounts. See “All Intersections Valid” on page 270.
- Some intersections are valid and others invalid for the source and destination accounts. See “Invalid Intersections” on page 270.
- One member is valid for the source account. See “One Valid Member on the Right Side” on page 271.

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 61:

<table>
<thead>
<tr>
<th>Member</th>
<th>Data in SourceAcct Intersection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member001</td>
<td>10</td>
</tr>
<tr>
<td>Member002</td>
<td>NoData status</td>
</tr>
</tbody>
</table>
All Intersections Valid

For the following example, all intersections of the TargAcct and SourceAcct accounts and the Custom1 members are valid:

\[ \text{HS.Exp } "A#TargAcct \Leftrightarrow A#SourceAcct" \]

The following table shows how this example puts data into the intersections of the TargAcct account and the Custom1 members:

<table>
<thead>
<tr>
<th>Custom1 Member</th>
<th>Data</th>
<th>Intersection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member001</td>
<td>10</td>
<td>SourceAcct and Member001</td>
</tr>
<tr>
<td>Member002</td>
<td>---</td>
<td>\text{Not applicable. The intersection of TargAcct and Member002 is unchanged because the intersection of SourceAcct and Member002 has a NoData status.}</td>
</tr>
<tr>
<td>Member003</td>
<td>15</td>
<td>SourceAcct and Member003</td>
</tr>
</tbody>
</table>

The following example uses Exp with the Member001 member on the left side of the equal sign:

\[ \text{HS.Exp } "A#TargAcct.C1#Member001 \Leftrightarrow A#SourceAcct" \]

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.

The following example uses EXP with Member003 on the right side of the equal sign:

\[ \text{HS.Exp } "A#TargAcct \Leftrightarrow A#SourceAcct.C1#Member003" \]

The following table shows how this example puts data into the intersections of the Custom1 members and the TargAcct account:

<table>
<thead>
<tr>
<th>Custom1 Member</th>
<th>Data</th>
<th>Intersection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member001</td>
<td>15</td>
<td>SourceAcct and Member003</td>
</tr>
<tr>
<td>Member002</td>
<td>15</td>
<td>SourceAcct and Member003</td>
</tr>
<tr>
<td>Member003</td>
<td>15</td>
<td>SourceAcct and Member003</td>
</tr>
</tbody>
</table>

Invalid Intersections

In the following examples, the source and destination accounts each have an invalid intersection. The valid and invalid intersections are described in the following list:

- SourceAcct. Member002 and Member003 are valid, and Member001 is invalid.
- TargAcct. Member001 and Member002 are valid, and Member003 is invalid.
In the following example, Exp is used without specifying a Custom1 member on either side of the equal sign:

\[ \text{HS.Exp } ^{\text{A#TargAcct = A#SourceAcct}} \]

Exp does not change data in the TargAcct account because Exp attempts to perform the following operations:

- \( \text{TargAcct.Member001 = SourceAcct.Member001} \). SourceAcct and Member001 is an invalid intersection.
- \( \text{TargAcct.Member002 = SourceAcct.Member002} \). Since the intersection of SourceAcct and Member002 has a NoData status, the intersection of TargAcct and Member002 remains unchanged.
- \( \text{TargAcct.Member003 = SourceAcct.Member003} \). TargAcct and Member003 is an invalid intersection.

In the following example, Exp is used with Member001 specified on the left side of the equal sign:

\[ \text{HS.Exp } ^{\text{A#TargAcct.Cl#Member001 = A#SourceAcct}} \]

TargAcct.Member001 remains unchanged because Exp attempts to retrieve data from an invalid intersection (SourceAcct and Member001).

In the following example, Exp is used with Member003 specified on the right side of the equal sign:

\[ \text{HS.Exp } ^{\text{A#TargAcct = A#SourceAcct.Cl#Member003}} \]

The following table shows how this example puts data into the intersections of the Custom1 members and the TargAcct account:

<table>
<thead>
<tr>
<th>Custom1 Member</th>
<th>Data</th>
<th>Intersection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member001</td>
<td>15</td>
<td>SourceAcct and Member003</td>
</tr>
<tr>
<td>Member002</td>
<td>15</td>
<td>SourceAcct and Member003</td>
</tr>
<tr>
<td>Member003</td>
<td>---</td>
<td>Not applicable. Member003 is an invalid intersection for the TargAcct account.</td>
</tr>
</tbody>
</table>

**One Valid Member on the Right Side**

In the following examples, the source account has only one valid member, and the destination account has two valid members. The valid and invalid intersections are described in the following list:

- SourceAcct. Member003 is the only valid intersection.
- TargAcct. Member001 and Member002 are valid, and Member003 is invalid.

In the following example, Exp is used without specifying a Custom1 member on either side of the equal sign:

\[ \text{HS.Exp } ^{\text{A#TargAcct = A#SourceAcct}} \]
The following table shows how this example puts data into the intersections of the Custom1 members and the TargAcct account:

<table>
<thead>
<tr>
<th>Custom1 Member</th>
<th>Data</th>
<th>Intersection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member001</td>
<td>15</td>
<td>SourceAcct and Member003 (the only valid intersection for the SourceAcct account)</td>
</tr>
<tr>
<td>Member002</td>
<td>15</td>
<td>SourceAcct and Member003 (the only valid intersection for the SourceAcct account)</td>
</tr>
<tr>
<td>Member003</td>
<td>---</td>
<td>Not applicable. Member003 is an invalid intersection for the TargAcct account.</td>
</tr>
</tbody>
</table>

In the following example, Exp is used with Member001 specified on the left side of the equal sign:

\[
\text{HS.Exp} \ "\text{A#TargAcct.C1#Member001 = A#SourceAcct}" \]

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 the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**

\[
\text{HS.GetCell}("\text{POVExpression}\")
\]

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

The following example assigns to the dData variable the amount stored in the intersection of the Sales account and the Golfballs member of the Custom1 dimension:

```plaintext
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 the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

Syntax

```plaintext
HS.GetCellNoData("POV", Var1)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POV</td>
<td>A valid point of view.</td>
</tr>
<tr>
<td>Var1</td>
<td>A variable that specifies if there is data in the cell.</td>
</tr>
</tbody>
</table>

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, then an error occurs.
Example

In the following example, the amount in the Sales account is assigned to the dSalesAmt variable. If the Sales account has no data, then 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 the following types of rules:

● Calculation
● Translation
● Consolidation
● Allocation

Syntax

HS.GetCellRealData("POV",Var1)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POV</td>
<td>A valid point of view.</td>
</tr>
<tr>
<td>Var1</td>
<td>A variable that specifies if there is real data in the cell.</td>
</tr>
</tbody>
</table>

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
GetCellType

Gets the cell type. This function can be used in the following 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

The following 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
```

GetItem

Gets an individual record to process for consolidation. This function can be used in Consolidation rules.
Syntax

HS.DataUnit.GetItem (lItem, strAccount, strICP, strCustom1, strCustom2, strCustom3, strCustom4, dData)

Table 64  Syntax for GetItem Function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lItem</td>
<td>A record number.</td>
</tr>
<tr>
<td>strAccount</td>
<td>Name of a valid Account dimension member.</td>
</tr>
<tr>
<td>strICP</td>
<td>Name of a valid ICP dimension member.</td>
</tr>
<tr>
<td>strCustom1</td>
<td>Name of a valid Custom1 dimension member.</td>
</tr>
<tr>
<td>strCustom2</td>
<td>Name of a valid Custom2 dimension member.</td>
</tr>
<tr>
<td>strCustom3</td>
<td>Name of a valid Custom3 dimension member.</td>
</tr>
<tr>
<td>strCustom4</td>
<td>Name of a valid Custom4 dimension member.</td>
</tr>
<tr>
<td>dData</td>
<td>The data in the specified cell.</td>
</tr>
</tbody>
</table>

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.

Syntax

HS.DataUnit.GetItemIDs2(lItem, lAccountID, lICPID, lCustom1ID, lCustom2ID, lCustom3ID, lCustom4ID, dData)

Table 65  Syntax for GetItemIDs2 Function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lItem</td>
<td>A record number.</td>
</tr>
<tr>
<td>lAccountID</td>
<td>ID number of a valid Account dimension member.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>lICPID</td>
<td>ID number of a valid ICP dimension member.</td>
</tr>
<tr>
<td>lCustom1ID</td>
<td>ID number of a valid Custom1 dimension member.</td>
</tr>
<tr>
<td>lCustom2ID</td>
<td>ID number of a valid Custom2 dimension member.</td>
</tr>
<tr>
<td>lCustom3ID</td>
<td>ID number of a valid Custom3 dimension member.</td>
</tr>
<tr>
<td>lCustom4ID</td>
<td>ID number of a valid Custom4 dimension member.</td>
</tr>
<tr>
<td>dData</td>
<td>The data in the specified cell.</td>
</tr>
</tbody>
</table>

**Return Value**

An array containing the account, ICP, Custom1...4, data.

**Example**

```vba
call DataUnit.GetItemIDs2(lItem, lAccount, lICP, lCustom1, lCustom2, lCustom3, lCustom4, 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**

```vba
NumItems = HS.DataUnit.GetNumItems
```

**Return Value**

The number of records in the data unit.

**Example**

```vba
set dataUnit = HS.OpenDataUnit(""
1NumItems = dataUnit.GetNumItems
for lItem = 0 to 1NumItems - 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 the following types of rules:
**Syntax**

`HS.GetNumLID("POVExpression")`

where `POVExpression` is a valid POV combination. If you do not specify a dimension, the following 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 the following 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.

```plaintext
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 following 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#2007.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

HS.GetSubmissionGroup
("A#Sales.C1#Golfballs.C2#Tennisballs.C3#Soccerballs.C4#Basketballs.I#EastSales")

GetSubmissionPhase

Gets the process management submission phase for the cell.

Syntax

HS.GetSubmissionPhase

Return Value

An integer representing the process management submission phase. Valid values are 1–9.
Example

HS.GetSubmissionPhase
#EastSales")

**Holding**

Gets the holding company for the current Entity dimension member or for a specified entity member. This function can be used in the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**

HS.Entity.Holding("Entity")
HS.Entity.Holding(""")

where *Entity* is the name of a valid Entity dimension 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.

**Example**

In the following 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 ICP Top Member of the current Account dimension member or the specified account member. This function can be used in the following 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 a sub consolidate routine.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccountName</td>
<td>Name of a valid Account dimension member.</td>
</tr>
<tr>
<td>Var1</td>
<td>A VisualBasic variable.</td>
</tr>
</tbody>
</table>

**Return Value**

A string with the name of the ICP Top Member for the account.

**Example**

In the following example, if the ICP Top Member of the Sales account is TotalProd, then statements between the If...Then and End If statements are executed.

```vbnet
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 the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**

HS.AppSettings.ICPWeight

**Return Value**

The percentage of ICP entities that are aggregated into the ICP top member. The value is a percentage scaled to hundreds, with 1.0 equalling 100 percent.
Example

In the following 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 the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

Syntax

HS.<Object>.IDFromMember("Element")

Table 67 Syntax for IDFromMember Function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Object&gt;</td>
<td>One of the following object keywords:</td>
</tr>
<tr>
<td></td>
<td>● Account</td>
</tr>
<tr>
<td></td>
<td>● Custom1...4</td>
</tr>
<tr>
<td></td>
<td>● Entity</td>
</tr>
<tr>
<td></td>
<td>● ICP</td>
</tr>
<tr>
<td></td>
<td>● Parent</td>
</tr>
<tr>
<td></td>
<td>● Period</td>
</tr>
<tr>
<td></td>
<td>● Scenario</td>
</tr>
<tr>
<td></td>
<td>● Value</td>
</tr>
<tr>
<td></td>
<td>● Year</td>
</tr>
<tr>
<td></td>
<td>● View</td>
</tr>
<tr>
<td>Element</td>
<td>Depending on the object selected, the name of a valid member of one of the following dimensions:</td>
</tr>
<tr>
<td></td>
<td>● Account</td>
</tr>
<tr>
<td></td>
<td>● Custom1...4</td>
</tr>
<tr>
<td></td>
<td>● Entity</td>
</tr>
<tr>
<td></td>
<td>● ICP</td>
</tr>
<tr>
<td></td>
<td>● Parent</td>
</tr>
<tr>
<td></td>
<td>● Period</td>
</tr>
<tr>
<td></td>
<td>● Scenario</td>
</tr>
</tbody>
</table>
Parameter | Description
---|---
● Value
● Year

Return Value
The ID number of the specified member.

Example
The following example gets the ID number for Connecticut:

```java
lEntityID = HS.Entity.IDFromMember("Connecticut")
```

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 68  Syntax for ImpactStatus Function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario</td>
<td>Name of a valid Scenario dimension member.</td>
</tr>
<tr>
<td>Year</td>
<td>A valid year.</td>
</tr>
<tr>
<td>Period</td>
<td>A valid period.</td>
</tr>
<tr>
<td>Entity</td>
<td>Name of a valid Entity dimension member.</td>
</tr>
<tr>
<td>Value</td>
<td>Name of a valid Value dimension member.</td>
</tr>
</tbody>
</table>

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.

Return Value
None.
**Input**

Enables data input into parent entities at base, calculated accounts. 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**

```plaintext
HS.Input("POVExpression")
```

where *POVExpression* is a point of view.

**Return Value**

None.

**Example**

The following example enables input into the cells that intersect the Sales account and the Budget scenario:

```plaintext
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 Financial Management epsilon. This function can be used in all types of rules.

**Syntax**

```plaintext
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 = 10.129999999
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 the following 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.<Object>.IsBase("", "")

Note:

Use a blank string (““) to apply this function to the current member.

Table 69 Syntax for IsBase Function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Object&gt;</td>
<td>One of the following object keywords:</td>
</tr>
<tr>
<td></td>
<td>● Account</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>● Custom1...4</td>
<td></td>
</tr>
<tr>
<td>● Entity</td>
<td></td>
</tr>
<tr>
<td>● Parent</td>
<td></td>
</tr>
</tbody>
</table>

**Parent**

A valid Parent member.

**Note:** Parent is mandatory only when used with Node.

**Element**

Depending on the object selected, the name of a valid member of one of the following dimensions:

- ● Account
- ● Custom1...4
- ● Entity
- ● Parent

**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 the following 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
```

**IsCalculated**

Determines if the current Account dimension member or a specified account member is a calculated account. This function can be used in the following 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 a sub consolidate routine.

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 the following 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 the following 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")
**Table 70  Syntax for IsChild Function**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| `<Object>` | One of the following object keywords:  
  - Account  
  - Custom1...4  
  - Entity  
  - Parent |
| Parent | Name of a Parent member. (Parent is mandatory.) |
| Element | Depending on the object selected, the name of a valid member of one of the following dimensions:  
  - Account  
  - Custom1...4  
  - Entity  
  - Parent |
| 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 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 the following example, if Connecticut is a child of EastRegion, then statements between the If...Then and End If lines are executed.

```plaintext
If HS.Entity.IsChild("EastRegion","Connecticut") = TRUE Then
```
IsConsolidated

Determines if the current Account dimension member or a specified account member is a consolidated account. This function can be used in the following 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 a sub consolidate routine.

**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 the following 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 the following 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 the following hierarchy, FosterCity and Sunnyvale are descendants of California and UnitedStates.

```
- UnitedStates
  - California
    - Sunnyvale
    - FosterCity
```

**Syntax**

```
HS.<Object>.IsDescendant("Parent","Element")
HS.Node.IsDescendant("Parent","Entity","S#Scenario.Y#Year.P#Period")
HS.<Object>.IsDescendant("Parent","")
```

**Note:**

Use a blank string (""") to apply this function to the current member.

**Table 71  Syntax for IsDescendant Function**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Object&gt;</td>
<td>One of the following object keywords:</td>
</tr>
<tr>
<td></td>
<td>● Account</td>
</tr>
<tr>
<td></td>
<td>● Custom1...4</td>
</tr>
<tr>
<td></td>
<td>● Entity</td>
</tr>
<tr>
<td></td>
<td>● Parent</td>
</tr>
<tr>
<td>Parent</td>
<td>Name of a valid Parent member. Parent is required.</td>
</tr>
<tr>
<td>Element</td>
<td>Depending on the object selected, name of a valid member of one of the following dimensions:</td>
</tr>
<tr>
<td></td>
<td>● Account</td>
</tr>
<tr>
<td></td>
<td>● Custom1...4</td>
</tr>
<tr>
<td></td>
<td>● Entity</td>
</tr>
<tr>
<td></td>
<td>● Parent</td>
</tr>
<tr>
<td>Entity</td>
<td>Name of a valid Entity dimension member.</td>
</tr>
<tr>
<td>Scenario</td>
<td>Name of a valid Scenario dimension member.</td>
</tr>
<tr>
<td>Year</td>
<td>A valid year.</td>
</tr>
<tr>
<td>Period</td>
<td>A valid period.</td>
</tr>
</tbody>
</table>
**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 the following example, if Connecticut is a descendant of Regional, then statements between the If...Then and End If lines are executed.

```hs
If HS.Entity.IsDescendant("Regional","Connecticut") = TRUE Then
    ... 
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 the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**

```hs
HS.<Object>.IsFirst
```

where `<Object>` is one of the following 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 the following 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 the following 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 72 Syntax for IsICP Function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Object&gt;</td>
<td>One of the following object keywords:</td>
</tr>
<tr>
<td></td>
<td>● Account</td>
</tr>
<tr>
<td></td>
<td>● Entity</td>
</tr>
<tr>
<td>Element</td>
<td>Depending on the object selected, name of a valid member of the Account or Entity dimension.</td>
</tr>
<tr>
<td></td>
<td>● Account</td>
</tr>
<tr>
<td></td>
<td>● Entity</td>
</tr>
</tbody>
</table>

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 the following 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

...
**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 the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**

```
HS.<Object>.IsLast
```

where `<Object>` is one of the following 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 the following 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 Translation 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 the following 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 Translation 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 the following 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 POV is a valid destination. This function can be used in the following
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, the following default values are used:

- Account - Current Account member if used in Sub Consolidate routine. Otherwise, account is required.
- Custom and ICP - Current member is used in Sub Consolidate routine. Otherwise, top member 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 POV is a valid destination; False otherwise.

Example
In the following example, if the specified destination is valid, then all statements between the If...Then and End If statements are executed.

```vbnet
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 Financial Management epsilon. This function can be used in all types of rules.

**Syntax**

```vbnet
BooleanValue = HS.IsZero(Value)
```

**Return Value**

A Boolean that is True if the passed in value is close to zero. False otherwise.

**Example**

```vbnet
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 the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

Syntax

HS.<Object>.List("Parent","Listname")
HS.Node.List("Parent","Listname","S#Scenario.Y#Year.P#Period")

Table 73 Syntax for List Function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Object&gt;</td>
<td>One of the following object keywords:</td>
</tr>
<tr>
<td></td>
<td>● Account</td>
</tr>
<tr>
<td></td>
<td>● Custom1...4</td>
</tr>
<tr>
<td></td>
<td>● Entity</td>
</tr>
<tr>
<td></td>
<td>● Parent</td>
</tr>
<tr>
<td></td>
<td>● ICP</td>
</tr>
<tr>
<td></td>
<td>● Scenario</td>
</tr>
<tr>
<td>Parent</td>
<td>Name of a valid Parent member.</td>
</tr>
<tr>
<td>Listname</td>
<td>Name of a valid system list or user-defined list.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: For node, must be a valid entity system list.</td>
</tr>
<tr>
<td>Scenario</td>
<td>Name of a valid Scenario dimension member.</td>
</tr>
<tr>
<td>Year</td>
<td>A valid year.</td>
</tr>
<tr>
<td>Period</td>
<td>A valid period.</td>
</tr>
</tbody>
</table>

Return Value

An array that contains all elements of the specified list. For Node, only the active elements in the list.

Example

The following example gets the elements of the user-defined list MyBaseList for the current account:

HS.Account.List("","MyBaseList")

The following example gets the elements of the system list [Base] for the TotalAssets account:
Member

Gets the name of the current member. This function can be used in the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

Syntax

HS.<Object>.Member

where <Object> is one of the following keywords:

- Entity
- Parent
- Period
- Scenario
- Value
- Year
- View

Return Value

A String that contains the name of the current member.

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 the following 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 the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**

```
HS.<Object>.MemberFromID(ElementID)
```

**Table 74 Syntax for MemberFromID Function**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;Object&gt;</code></td>
<td>One of the following object keywords:</td>
</tr>
<tr>
<td></td>
<td>● Account</td>
</tr>
<tr>
<td></td>
<td>● Custom1...4</td>
</tr>
<tr>
<td></td>
<td>● Entity</td>
</tr>
<tr>
<td></td>
<td>● ICP</td>
</tr>
<tr>
<td></td>
<td>● Parent</td>
</tr>
<tr>
<td></td>
<td>● Period</td>
</tr>
<tr>
<td></td>
<td>● Scenario</td>
</tr>
<tr>
<td></td>
<td>● Value</td>
</tr>
<tr>
<td></td>
<td>● Year</td>
</tr>
<tr>
<td></td>
<td>● View</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><code>Parent</code></th>
<th>A valid Parent member.</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ElementID</code></td>
<td>Depending on the object selected, the ID number of a valid member of one of the following dimensions:</td>
</tr>
<tr>
<td></td>
<td>● Account</td>
</tr>
<tr>
<td></td>
<td>● Custom1...4</td>
</tr>
<tr>
<td></td>
<td>● Entity</td>
</tr>
<tr>
<td></td>
<td>● ICP</td>
</tr>
<tr>
<td></td>
<td>● Parent</td>
</tr>
<tr>
<td></td>
<td>● Period</td>
</tr>
<tr>
<td></td>
<td>● Scenario</td>
</tr>
<tr>
<td></td>
<td>● Value</td>
</tr>
<tr>
<td></td>
<td>● Year</td>
</tr>
</tbody>
</table>

**Return Value**

The name of the dimension member.
**Example**

The following example gets the member for the ID number 001:

```csharp
strEntity = HS.Entity.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 the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**

```csharp
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 the following example, if the method for the point of view is GLOBAL, then statements between the If...Then and End If statements are executed.

```csharp
If HS.Node.Method("S#Actual","Y#2007","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**

```csharp
HS.NoInput "POVExpression"
```
where \textit{POVExpression} is a point of view.

\textbf{Return Value}

None.

\textbf{Example}

The following example prohibits input into the cells that intersect the Sales account and the Budget scenario for 2007:

\begin{verbatim}
Sub NoInput
    HS.NoInput "S#Budget.Y#2007.A#Sales"
End Sub
\end{verbatim}

\textbf{NoRound}

Turns off rounding for all following \texttt{Exp} statements. This function can be used in the following types of rules:

\begin{itemize}
  \item Calculation
  \item Translation
  \item Allocation
\end{itemize}

\textbf{Tip:}

You can also turn off rounding by entering 0 as the argument for the \texttt{Round} function. For example, \texttt{HS.Round(0)} turns off rounding.

\textbf{Syntax}

\texttt{HS.NoRound}

\textbf{Return Value}

None.

\textbf{Example}

The following 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:

\begin{verbatim}
HS.Round 0.1
HS.Exp "A#SalesRound" = "A#Sales"
HS.NoRound
HS.Exp "A#SalesNoRound" = "A#Sales"
\end{verbatim}
**NumBase**

Gets the number of base members for the current member or for a specified member. This function can be used in the following 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**

```plaintext
HS.<Object>.NumBase("Element")
HS.Node.NumBase("S#Scenario.Y#Year.P#Period.E#Entity")
HS.<Object>.NumBase("")
```

**Note:**
Use a blank string ("") to apply this function to the current member.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Object&gt;</td>
<td>One of the following object keywords:</td>
</tr>
<tr>
<td></td>
<td>● Account</td>
</tr>
<tr>
<td></td>
<td>● Custom1...4</td>
</tr>
<tr>
<td></td>
<td>● Entity</td>
</tr>
<tr>
<td></td>
<td>● Parent</td>
</tr>
<tr>
<td></td>
<td>● Node</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Element</th>
<th>Depending on the object selected, name of a valid member for one of the following dimensions:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>● Account</td>
</tr>
<tr>
<td></td>
<td>● Custom1...4</td>
</tr>
<tr>
<td></td>
<td>● Entity</td>
</tr>
<tr>
<td></td>
<td>● Parent</td>
</tr>
<tr>
<td></td>
<td>● Node</td>
</tr>
</tbody>
</table>

For Account and Custom objects, you must specify the member - you cannot use a blank string.

To get the number of base members in the entire dimension, specify `ALL` within quotation marks, as in the following example:

```
iAcctBase = HS.Account.NumBase("ALL")
```
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario</td>
<td>Name of a valid Scenario dimension member.</td>
</tr>
<tr>
<td>Year</td>
<td>A valid year.</td>
</tr>
<tr>
<td>Period</td>
<td>A valid period.</td>
</tr>
<tr>
<td>Entity</td>
<td>Name of a valid Entity dimension member.</td>
</tr>
</tbody>
</table>

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 the following example:

```hs
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 the following 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.

```hs
If HS.Exp"A#SalesAlloc = A#Sales/HS.Entity.NumBase(Regional)" then
...
End If
```

**Number**

Gets the current period number. This function can be used in the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**

```hs
HS.View.PeriodNumber
```
Return Value
The current period number.

Example
In the following 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 the following 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(""")

Note:
Use a blank string ("") to apply this function to the current member.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Object&gt;</td>
<td>One of the following object keywords:</td>
</tr>
<tr>
<td></td>
<td>● Account</td>
</tr>
<tr>
<td></td>
<td>● Custom1-4</td>
</tr>
<tr>
<td></td>
<td>● Entity</td>
</tr>
<tr>
<td></td>
<td>● Node</td>
</tr>
<tr>
<td></td>
<td>● Parent</td>
</tr>
</tbody>
</table>
### Parameter | Description
---|---
**Element** | Depending on the object selected, name of a valid member of one of the following dimensions:
- Account
- Custom1...4
- Entity
- Node
- Parent
For Account and Custom objects, you must specify the member - you cannot use a blank string.
To get the number of child members in the entire dimension, specify **ALL** within quotation marks, as in the following example:
```
iAcctBase = HS.Account.NumChild("ALL")
```
**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 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 Long that identifies the number of child members. For Node, gets the number of active children of the specified member.

#### Example
In the following 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)"
```

### NumDescendant

Gets the number of descendants of the current dimension member or a specified member. This function can be used in the following 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 the following hierarchy, FosterCity and Sunnyvale are descendants of California and UnitedStates.

```
- UnitedStates
  - California
    - Sunnyvale
    - FosterCity
```

**Syntax**

```
HS.<Object>.NumDescendant("Element")
HS.Node.NumDescendant("S#Scenario.Y#Year.P#Period.E#Entity")
HS.<Object>.NumDescendant(""")
```

**Note:**

Use a blank string ("") to apply this function to the current member.

**Table 77  Syntax for NumDescendant Function**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| `<Object>` | One of the following object keywords:  
  - Account  
  - Custom1-4  
  - Entity  
  - Node  
  - Parent |
| `Element` | Depending on the object selected, name of a valid member of one of the following dimensions:  
  - Account  
  - Custom1...4  
  - Entity  
  - Node  
  - Parent  
  
  For Account and Custom objects, you must specify the member - you cannot use a blank string.  
  
  To get the number of descendant in the entire dimension, specify **ALL** within quotation marks, as the argument as in the following example:  
  
  ```
iAcctBase = HS.Account.NumDescendant("ALL")
```  

| `Scenario` | Name of a valid Scenario dimension member. |
| `Year` | A valid year. |
| `Period` | A valid period. |
### Parameter | Description
--- | ---
**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 the following example, if the entity France has no descendants, then statements between the `If...Then` and `End If` statements are executed.

```plaintext
If HS.Entity.NumDescendant("France") = 0 then
  ...
End If
```

### OpenDataUnit

Gets the data unit to process during consolidation, calculation, or translation. This function can be used in the following types of rules:

- Calculation
- Translation
- Consolidation

#### Syntax

```plaintext
Set DataUnit = HS.OpenDataUnit(POVExpression)
```

where `POVExpression` is a POV.

#### Return Value
When used in a sub consolidate routine, returns all records with data but only returns accounts that are flagged as consolidated.

When used in a sub calculate or sub translate routine, 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

**PCon**

Gets the percentage of consolidation for the current member or a specified member. This function can be used in the following types of rules:

- **Calculation**
- **Translation**
- **Consolidation**
- **Allocation**

**Syntax**

```plaintext
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 78 Syntax for PCon Function**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario</td>
<td>Name of a valid Scenario dimension member.</td>
</tr>
<tr>
<td>Year</td>
<td>A valid year.</td>
</tr>
<tr>
<td>Period</td>
<td>A valid period.</td>
</tr>
<tr>
<td>Entity</td>
<td>Name of a valid Entity dimension member.</td>
</tr>
</tbody>
</table>

**Return Value**

The percentage of consolidation for the member.

**Example**

The following example gets the percent consolidation for the specified point of view:

```plaintext
```

**PlugAcct**

Gets the plug account for the current Account member or for a specified account. This function can be used in the following types of rules:
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 a sub consolidate routine.

Return Value
A string that specifies the name of the plug account for the member.

Example
In the following 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 the following 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 79  Syntax for POwn Function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario</td>
<td>Name of a valid Scenario dimension member.</td>
</tr>
<tr>
<td>Year</td>
<td>A valid year.</td>
</tr>
<tr>
<td>Period</td>
<td>A valid period.</td>
</tr>
<tr>
<td>Entity</td>
<td>Name of a valid Entity dimension member.</td>
</tr>
</tbody>
</table>

Return Value
The percentage of ownership for the member.

Example
The following example gets the percent ownership for the specified point of view:


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 the following 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 the following 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 the following 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 the following 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 the following 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 the following 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 the following 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 the following 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 the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation
Syntax

HS.ReviewStatus("$Scenario.Y#Year.P#Period.E#Entity.V#Value$")
HS.ReviewStatus(""")

Note:
Use a blank string ("") to apply this function to the current member.

Table 80  Syntax for ReviewStatus Function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario</td>
<td>Name of a valid Scenario dimension member.</td>
</tr>
<tr>
<td>Year</td>
<td>A valid year</td>
</tr>
<tr>
<td>Period</td>
<td>A valid period.</td>
</tr>
<tr>
<td>Entity</td>
<td>Name of a valid Entity dimension member.</td>
</tr>
<tr>
<td>Value</td>
<td>Name of a Value dimension member.</td>
</tr>
</tbody>
</table>

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 the following 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


Round

Rounds data from the Exp function. You specify the degree of rounding in the argument. This function can be used in the following 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. The following 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
The following 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 the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

Syntax

```
HS.Currency.Scale("Currency")
HS.Currency.Scale(Var1)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Currency</td>
<td>Name of a valid currency.</td>
</tr>
<tr>
<td>Var1</td>
<td>A VisualBasic variable.</td>
</tr>
</tbody>
</table>

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 the following 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 when the entity is used as an intercompany partner. This function can be used in the following types of rules:

● Calculation
● Translation
● Consolidation
● Allocation

Syntax

HS.Entity.SecurityAsPartner("Entity")
HS.Entity.SecurityAsPartner(""")
HS.Entity.SecurityAsPartner(Var1)

Note:
Use a blank string (""") to apply this function to the current entity member.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity</td>
<td>Name of a valid Entity dimension member.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>Var1</td>
<td>A VisualBasic variable.</td>
</tr>
</tbody>
</table>

**Return Value**

A string with the security class assigned to the entity when it is used as an ICP.

**Example**

In the following 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:

```plaintext
If HS.Entity.SecurityAsPartner("France") = "Class1" Then
    ...
End If
```

---

**SecurityClass**

Gets the security class for the specified member. This function can be used in the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**

```plaintext
HS.<Object>.SecurityClass("Element")
HS.<Object>.SecurityClass("")
HS.<Object>.SecurityClass(Var1)
```

**Note:**

Use a blank string (" ") to apply this function to the current member.

**Table 83 Syntax for SecurityClass Function**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Object&gt;</td>
<td>One of the following object keywords:</td>
</tr>
<tr>
<td></td>
<td>● Account</td>
</tr>
<tr>
<td></td>
<td>● Scenario</td>
</tr>
<tr>
<td></td>
<td>● Entity</td>
</tr>
<tr>
<td></td>
<td>● Custom1...4</td>
</tr>
<tr>
<td>Element</td>
<td>Depending on the object selected, name of a valid member of one of the following dimensions:</td>
</tr>
</tbody>
</table>
Parameter | Description
---|---
● Account
● Scenario
● Entity
● Custom1...4

Var1 | A VisualBasic variable.

Return Value
The name of the security class assigned to the specified member.

Example
In the following example, if Class1 is the security class assigned to the Cash account, then statements between the If...Then and End If statements are executed:

```vbnet
If HS.Account.SecurityClass("Cash") = "Class1" Then
    ...
End If
```

**SetData**

Sets an individual record. This function can be used in the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation

**Syntax**

```vbnet
HS.SetData lView, lAccount, lICP, lCustom1, lCustom2, lCustom3, lCustom4, dData, bAddToExistingData
```

**Table 84 Syntax for SetData Function**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| lView     | 0 = Scenario View  
                1 = Periodic  
                2 = YTD |
| lAccount  | ID number of the account to which you are setting data. |
| lICP      | ID number of the ICP to which you are setting data. |
| lCustom1...4 | ID number of the Custom dimension to which you are setting data. |
| dData     | The data value to set. |
### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bAddToExistingData</td>
<td>True = To accumulate the data</td>
</tr>
<tr>
<td></td>
<td>False = To replace the data</td>
</tr>
</tbody>
</table>

**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 the following types of rules:

- Calculation
- Translation
- Consolidation
- Allocation
- Dynamic Calculations
- Transactions

**Syntax**

```
HS.SetDataWithPOV POV, dData, bAddToExistingDataInCache
```

**Table 85 Syntax for SetData Function**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POV</td>
<td>Valid POV</td>
</tr>
<tr>
<td>dData</td>
<td>The data value to set.</td>
</tr>
<tr>
<td>bAddToExistingDataInCache</td>
<td>True = To accumulate the data</td>
</tr>
<tr>
<td></td>
<td>False = To replace the data</td>
</tr>
</tbody>
</table>

**Return Value**

None.

**Example**

```
HS.SetDataWithPOV "V#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 the following dimensions: Account, Custom1...4.

HS.Account.SubmissionGroup(Account)
HS.Custom1.SubmissionGroup(Custom1)
HS.Custom2.SubmissionGroup(Custom2)
HS.Custom3.SubmissionGroup(Custom3)
HS.Custom4.SubmissionGroup(Custom4)

**Return Value**

An integer representing the submission group for the dimension member. Valid values are 1–99.

**Example**

HS.Account.SubmissionGroup("Sales")

SupportsTran

Specifies the accounts in the application that support intercompany transactions. This function can be used 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.Cl#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 the following 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 the following types of rules:
• Calculation
• Translation
• Consolidation
• Allocation

Syntax

HS.<Object>.SwitchSign("Member")
HS.<Object>.SwitchSign(""")

Note:
Use a blank string (""") to apply this function to the current member.

Table 86 Syntax for SwitchSign Function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| <Object>  | One of the following object keywords:  
|           | ● Custom1   
|           | ● Custom2   
|           | ● Custom3   
|           | ● Custom4   |
| Member    | 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 the following 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
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 the following 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 the following 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 87  Syntax for SwitchType Function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Object&gt;</td>
<td>One of the following object keywords:</td>
</tr>
<tr>
<td></td>
<td>● Custom1</td>
</tr>
<tr>
<td></td>
<td>● Custom2</td>
</tr>
<tr>
<td></td>
<td>● Custom3</td>
</tr>
<tr>
<td></td>
<td>● Custom4</td>
</tr>
<tr>
<td>Member</td>
<td>Name of a valid Custom dimension member.</td>
</tr>
</tbody>
</table>

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 the following example, if the account types are switched for the current Custom1 member, then statements between the If..Then and End If statements are executed:

```plaintext
If HS.Custom1.SwitchType("") = "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>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>DestPOV</em></td>
<td>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.</td>
</tr>
<tr>
<td><em>SourcePOV</em></td>
<td>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 &lt;EntityCurrTotal&gt; member. If the source point of view is blank, the system uses the destination point of view as the source point of view.</td>
</tr>
<tr>
<td><em>Rate1-2</em></td>
<td>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 “Default Translation” on page 221.</td>
</tr>
</tbody>
</table>

**Return Value**

None.

**Example**

The following 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 89 Syntax for TransPeriodic Function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DestPOV</td>
<td>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.</td>
</tr>
<tr>
<td>SourcePOV</td>
<td>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 &lt;EntityCurrTotal&gt; member. If the source is blank, the system uses the destination as the source.</td>
</tr>
<tr>
<td>Rate1-2</td>
<td>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 “Default Translation” on page 221.</td>
</tr>
</tbody>
</table>

Return Value
None

Example
The following 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 the following types of rules:
- Calculation
- Translation
- Consolidation
- Allocation

Syntax
HS.<Object>.UD1...3(strElement)
HS.<Object>.UD1...3(""")

Note:
Use a blank string (""") to apply this function to the current member.
Table 90  Syntax for UD1...3 Functions

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| <Object>  | One of the following object keywords:  
|           | ● Account  
|           | ● Entity   
|           | ● Parent   
|           | ● Scenario 
|           | ● Custom1...4 |
| Element   | Depending on the object selected, name of a valid member of one of the following dimensions:  
|           | ● Account  
|           | ● Entity   
|           | ● Parent   
|           | ● Scenario 
|           | ● Custom1...4 |

Return Value

A String that contains the user-defined text stored for the member.

Example

In the following 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

ValidationAccount

Gets the validation account for an application. This function can be used in the following 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
The following 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 the following 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 91  Syntax for XBRLTags Functions

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account</td>
<td>A valid account.</td>
</tr>
</tbody>
</table>

Return Value

A string that specifies the XBRL tag for the specified account.
Managing Intercompany Transaction Periods

Before you can enter, load, or process intercompany transactions, you must first open the period for the transactions. After you have finished processing transactions, you can close the period to prevent further input.

For more information, see these procedures:

- “Opening Periods” on page 327
- “Closing Periods” on page 329

Opening Periods

You open a period by specifying the scenario, year, and period. 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 go back to Unopened.

When you open periods, you can set the Match/Validate Before Post option and matching tolerances that apply to the Auto-Match and Manual Match processes.

To open periods:

1. In the Browser View, expand Tasks and select IC Transaction Tasks.
2. Select Manage IC Periods.
3. From Scenario, select a scenario for the period.
4. From Year, select a year for the period.
5. Select the check box next to each period to open.
6 Optional: In the Matching Tolerance column, enter a TID, Account, or Manual matching tolerance for the period.

7 In the Match/Validate Before Post column, select one of the following options:
   - If you require the system to check the match status before posting transactions, select Yes or Restricted.
   
   **Note:**
   If set to Yes, additional validation is done in the close period and lock entity processes.
   - If you do not require the system to check the match status, select No.

8 Click Open Period.

9 Optional: To save the settings for the period, click Save Period Settings.

### Setting Matching Tolerances


For account and manual match, 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.

### Setting Match/Validate Before Post Option

When you open periods, 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.

The Match/Validate Before Post option can be set to Yes, No, or Restricted.

<table>
<thead>
<tr>
<th>Match/Validate Option Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>All transactions can be posted</td>
</tr>
</tbody>
</table>
| Yes                         | The system allows transactions to be posted if either of the following conditions are met:  
|                             | - Transactions have the Matched status.  
|                             | - Transactions with a MisMatched status contain a valid Reason Code.  
|                             | **Note:** Unmatched transactions or mismatched transactions without a reason code cannot be posted. |
### Match/Validate Option Value

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The system also checks that all matched transactions or mismatched transactions with reason code are posted before the period can be closed or the entity can be locked.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Restricted</th>
</tr>
</thead>
<tbody>
<tr>
<td>If the Match/Validate Before Posted option is set to Restricted, the system allows transactions to be posted if either of the following conditions are met:</td>
</tr>
<tr>
<td>● Transactions have the Matched status.</td>
</tr>
<tr>
<td>● Transactions with a Mismatched status contain a valid Reason Code.</td>
</tr>
<tr>
<td><strong>Note:</strong> Unmatched transactions or mismatched transactions without a reason code cannot be posted.</td>
</tr>
<tr>
<td>The system does not require that all matched transactions or mismatched transactions with reason code be posted before the period can be closed or the entity can be locked.</td>
</tr>
</tbody>
</table>

### Closing Periods

At the end of the closing cycle when you have finished processing intercompany transactions, you can close the period to prevent additional modifications to the transactions. If the Match/Validate Before Post option is set to Yes for the period, all matched transactions or mismatched transactions with a reason code must be posted before a period can be closed. If Match/Validate Before Post is set to No or Restricted, the system does not check before closing the period.

When you close a period, it is locked for future transactions, however you can continue to view transactions or run reports on the period’s transactions.

**To close a period:**

1. In the **Browser View**, expand **Tasks** and select **IC Transaction Tasks**.
2. Select **Manage IC Periods**.
3. From the **Scenario** drop-down list, select a scenario for which to close periods.
4. From the **Year** drop-down list, select a year in which to close periods.
5. Select the check box next to each period to close.
6. Click **Close Period**.

### 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 set to Yes for the period, all matched transactions or mismatched transactions with a reason code must be posted before the entity can be locked. If the Match/Validate Before Post option is set to No or Restricted, the system does not check before locking the period.
Note:
The transaction lock status is different from the data lock status. For information on data lock status, see the Hyperion Financial Management — System 9 User’s Guide.

When the entity is locked, you are not allowed to enter any new intercompany transactions. You are also not allowed to delete or make any changes to existing transactions for the entity. You cannot post or unpost any transactions to a locked entity, or update the match status of the entity’s transaction. Therefore, even if the partner entity is not locked, the partner cannot match their transactions with the entity’s 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. In the Browser View, expand Tasks and select IC Transaction Tasks.
2. Select Lock/Unlock IC Entities.
3. From Scenario, select a scenario for the entity.
4. From Year, select a year for the entity.
5. From Period, select a period for the entity.
6. Select the check box next to each entity to lock.
7. Click Lock.

Tip:
To unlock entities, select the check box next to each entity to unlock, then click Unlock.

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.

For more information, see these procedures:
- “Adding Reason Codes” on page 331
- “Editing Reason Codes” on page 331
- “Removing Reason Codes” on page 332

## 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 *Hyperion Financial Management — System 9 User’s Guide*.

➤ To add reason codes:

1. In the Browser View, expand Tasks and select IC Transaction Tasks.
2. Select Manage IC Reason Codes.
3. For New Reason Code, 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.
4. For Description, enter a description for the reason code.
   
   **Note:**
   The description can contain a maximum of 40 characters.
5. Click Add.

## Editing Reason Codes

After you create a reason code, you can edit the code description.

➤ To edit reason codes:

1. In the Browser View, expand Tasks and select IC Transaction Tasks.
2. Select Manage IC Reason Codes.
3. From the list of reason codes, select the reason code to edit.
4 Click Edit.
5 From the Description column, edit the description, then click OK.

**Removing Reason Codes**

You can remove reason codes that you no longer need in the list of reason codes for an application.

➤ To remove reason codes:

1 In the Browser View, expand Tasks and select IC Transaction Tasks.
2 Select Manage IC Reason Codes.
3 From the list of reason codes, select the reason code to remove.
4 Click Remove.
5 Click Yes to remove the reason code.

**Monitoring Intercompany Transactions**

You can 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.

The Intercompany Transactions Monitor window displays a list of intercompany entities with their Process status and Lock status. The entities are links to Intercompany Transactions Monitor Detail information. When you click an entity, the system opens a pop-up window that displays the number of posted and unposted transactions by status, such as Matched, Mismatched, or Unmatched. For more information, see “Viewing the Intercompany Transactions Summary” on page 334.

The values in the Intercompany Transactions Monitor Detail window link to the Process IC Transactions window with the filtering to display the detail transactions from the values.

For example, if you click Entity A from the list of entities, the Intercompany Transactions Monitor Detail pop-up window 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 Process IC Transactions window, with the filtering selected to display the unposted transactions with an Unmatched status for Entity A.

You can send email alerts for any of the entities in the Intercompany Transactions Monitor window. For more information on email alerting, see the *Hyperion Financial Management — System 9 User's Guide*. 
**Note:**

To monitor intercompany transactions, you must be assigned the Intercompany Transaction Administrator security role.

You can monitor the status of intercompany transactions between entities and intercompany partners. The system displays the process status and lock status of the intercompany transactions for the selected entities, and you can filter and sort the list by status.

➤ To monitor intercompany transactions:

1. In the Browser View, expand Tasks and select IC Transaction Tasks.
2. Select Monitor IC Transactions.
3. In the Point of View bar, select a scenario, year, and period.
4. For Entity, enter or browse for the entity for which to monitor intercompany transaction status.

**Note:**

If you leave Entity blank, the system returns all entities in the list.

5. From the Display drop-down list, select one of the following options:
   - 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.

6. **Optional:** To filter the transactions list by process or lock status, select one or more of the following transaction types, then click Update to refresh the list:
   - Not Started
   - Started
   - Not Lockable
   - Lockable
   - Locked

7. **Optional:** To sort the transactions list by Process status, click Sort to sort by Ascending or Descending.
8. **Optional:** To sort the transactions list by Lock status, click Sort to sort by Ascending or Descending.
9. Click an entity to link to the Intercompany Transactions Monitor Detail information.
10. From the Intercompany Transactions Monitor Detail window, click a value from one of the status columns. The Process IC Transactions window is displayed with the filtering selected.
11. When you finish viewing monitor detail, from the Intercompany Transactions Monitor Detail window, click Close Window.
**Viewing the Intercompany Transactions Summary**

You can view a summary of the status of the intercompany transactions for a specific entity based on the selections displayed in the Monitor Intercompany Transactions window.

➤ To view the intercompany transactions summary:

1. From the Monitor Intercompany Transactions window, select the entity for which to view a summary of transactions.
2. When you finish viewing the summary, click Close Window.
Managing Shared Services Models

In This Chapter

Prerequisites ........................................335
Registering Applications.................................335
About Models ........................................335
About Managing Models .................................336
About Sharing Metadata and Sharing Data ........336
Working with Applications ...............................336
Working with Models ..................................337

Prerequisites

Shared Services supports external directories for user authentication. To use Shared Services functionality, you must configure Shared Services and Financial Management to use external authentication. See the Hyperion Security Administration Guide.

Registering Applications


About Models

Shared Services provides a database, organized into applications, in which applications can store and manage metadata models. A model is a container of application-specific data, such as a file or string. There are two types of models: dimensional hierarchies such as entities and accounts, and nondimensional objects such as security files, member lists, rules, scripts, and data forms.

Some Hyperion products require that models be displayed within a folder structure (similar to Windows Explorer). Folder views enable the administrator to migrate an entire folder structure or a portion of a folder structure easily using Shared Services.
The process of copying a model or folder from a local application to Shared Services is known as exporting.

**About Managing Models**

Shared Services enables you to store metadata models and application folders. A separate application is provided for each product.

Shared Services provides basic management functionality for models and folders:

- Version tracking
- Access control
- Ability to rename and delete models

Users must be assigned the Manage Models user role to perform the preceding actions on Shared Services models.

**Note:**

The Manage Models user must have Manage permission for a model using the Shared Services Model Access window to assign permissions to it.

For information about assigning user roles, see the *Hyperion Security Administration Guide*.

**About Sharing Metadata and Sharing Data**

For Hyperion products that support Performance Management Architect, the metadata sharing and data sharing functionality is no longer supported through Shared Services. Instead it is supported through Oracle’s Enterprise Performance Management Architect.

**Working with Applications**

Metadata models are stored in directories in Shared Services. Shared Services manages models at the application level. Each application that is registered with Shared Services has a corresponding application in Shared Services, in which it stores its Shared Services models. An application has exclusive use of the application models.

To put a local copy of a model under control of Shared Services, you export the model to the application directory in Shared Services.

To manage models, Shared Services provides such management capabilities as:

- Track model versions
- Control access to models
Working with Models

You store and manage models in Shared Services. The Manage Models Browse tab lists the models in the product.

➤ To list the models in Shared Services:

1. From Financial Management, select Administration > Manage Models.
2. If it is not already selected, select Browse.

Note:

Some Hyperion products do not display folders in the Manage Models Browse tab.

Note:

If the current application is new, the view might not show models. Application models are displayed in the Browse tab after you explicitly export them to Shared Services.

All models are displayed in ascending order. The Manage Models Browse tab provides information about each model in Shared Services:

- Model name
- Model type
- Last time the model was updated
- Whether the model is locked and who locked it

You can see only the models to which you have at least Read access. If you do not have access to a model, it is not displayed in the Manage Models Browse tab.

Some Hyperion products require that models be displayed within a folder structure (similar to Windows Explorer). Folder views enable the administrator to migrate an entire folder structure or a portion of a folder structure easily using Shared Services. Folders are visible on the Manage Models Browse tab. Path information for folders is displayed directly above the column headers and path text is hyperlinked to refresh the page within the context of the selected folder.

From the Manage Models Browse tab, you can:

- View members and member properties in dimensional models.
- Compare the latest application version of a model to the latest version stored in Shared Services.
- Track model history.
- View model properties.
- Rename models.
- Delete models.
Exporting Models and Folders

The Manage Models window lists the latest version of each model in Shared Services. Shared Services also tracks models in the application and determines whether a version of each model resides in the Financial Management application only, in Shared Services only, or in both.

When the latest version of a model resides in both the Financial Management application and in Shared Services, the Financial Management application and Shared Services are said to be synchronized, or in sync, with regard to that model.

If a model is out of sync, you can synchronize it by exporting the model to Shared Services. You need Write permission to synchronize a model.

Note:

Models within folders can also be synchronized using the Shared Services sync operation. If a folder is selected, all models within that folder and within any subfolders will be synchronized.

To synchronize models and folders:

1. From Financial Management, select Administration > Manage Models.
2. Select Sync.

The Sync Preview window lists all models and folders in Shared Services and in the Financial Management application. The Sync Operation field provides a recommended operation to apply to each model or folder.

3. Optional: For models with Select Sync Operation, you can compare the latest version of the model in Shared Services to the model in the application by clicking Compare. Before clicking Compare, you must select a Sync Operation in the drop-down list.

The latest version of the model in Shared Services is compared to the latest version in the application. The contents of the two models are shown line-by-line in a side-by-side format. Shared Services Version refers to the model in Shared Services. Application Version refers to the model in the application.

After you resolve the differences in a model, you are returned to the Sync Preview page.

4. Select each model to synchronize.

For models with Select Sync Operation, select a Sync Operation depending on whether the application or Shared Services has the latest version of the model.

Note:

Before exporting a model from an application to Shared Services, verify that the model names do not violate Shared Services naming restrictions.

5. Synchronize the selected models.

a. Click Sync.

A window displays, enabling you to enter comments for each of the selected models.
b. Type comments for each model, or type comments for one model and click Apply To All.
c. Click Sync.

6 Click Report to see a report of the operations that have been completed.
7 Click Refresh to update the message.
8 Click Close to return to the Sync Preview window.

Export Operations

The Sync Preview window lists all models in Shared Services and in the application. The Sync Operation field provides a recommended operation to apply to each model:

● If a model exists in the application but not in Shared Services, the sync operation is Export to Shared Services. You cannot change this operation. If you select the model, when you synchronize, the specified model is copied to Shared Services.

Note:

Shared Services supports dimensions that contain up to 100,000 members.

● If a model exists in both the application and Shared Services, the sync operation is selectable. Select from one of these options:

Note:

With export, the compare operation considers the application model to be the “master” model.

○ Export with Merge — Merges the application model content with the content in Shared Services. Notice:
  □ If a property only exists in the application model, then the property is retained in the merged model.
  □ If a property only exists in the Shared Services model, then the property is retained in the merged model.
  □ If a property exists in both models, the value of the property in the application model is retained in the merged model.
  □ A member in the application model but not in the Shared Services model is retained in the merged model.
  □ A member in the Shared Services model but not in the application model is not retained in the merged model.
  □ A member which exists both in the Shared Services model and in the application model, but in different generation levels, is merged and the position in the application model is maintained.
If an application system member exists only in a Shared Services model, export with merge does not delete this member. If an application system member exists both in a Shared Services model and in the application model, export with merge will merge the properties as usual and take the system member-specific attributes from the application model.

For properties with attributes, the merge is based on the attribute value. For example, if the following “Alias” attribute exists in the Shared Services model:

```xml
<Alias table="French">Text in French</Alias>
```

and if the following “Alias” attribute exists in the application model:

```xml
<Alias table="English">Text in English</Alias>
```

then the merged result will contain both attributes and looks like this:

```xml
<Alias table="French">Text in French</Alias>
<Alias table="English">Text in English</Alias>
```

If the value for both “Alias” attributes is the same in both models, then the value for the application model is retained in the merged model.

Export with Overwrite — Replaces the Shared Services model with the application model.

## Model Naming Restrictions

Restrictions on model names:

- The maximum length is 80 characters, regardless of the application in which you are working.
- Names are not case sensitive.
- You can use all alphanumeric and special characters, with the exception of the forward slash (/) and double quotation (") characters. Therefore, you cannot export a dimension to Shared Services that contains forward slash or double quotation characters.

Note:

Financial Management may enforce additional restrictions on names.

## Comparing Models

You can compare a model in Shared Services to its corresponding version in the application. The latest version in Shared Services is compared to the model in the application.

To compare the application representation of a model with the Shared Services representation of the model:

1. From Financial Management, select Administration > Manage Models.
2 Select Sync.

The Sync Preview window lists all models in Shared Services and in the application. The Sync Operation field provides a recommended operation to apply to each model or folder.

3 Select a Sync Operation in the drop-down list for the model of interest.

4 Click Compare next to Sync Operation.

The latest version of the model in the application is compared with the latest version in Shared Services.

5 Perform any compare operations.

6 Click OK to return to the Sync Preview window.

**Compare Operations**

The contents of the two models are shown line-by-line in a side-by-side format. Application Version refers to the model in the application. Application versions of a model are displayed on the left side of the Resolve Models (Compare) window. Shared Services Version refers to the model in Shared Services. Shared Services versions of a model are displayed on the right side of the Compare Models window.

By default, the Resolve Models window displays up to 50 rows per page, displays any folders in an expanded format, and displays only those models with differences.

Color coding highlights any differences between the content of the two models:

- Red indicates that the element has been deleted from the model.
- Green indicates that the element has been inserted into the model.
- Blue indicates that the element has been changed.

**Note:**

The compare operation filters out any application system members that are not relevant to the product being viewed. For example, when you view Financial Management models, Shared Services filters out any application system members that are not valid for Financial Management.

**Table 93** describes the Resolve Models (Compare) window elements.

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expand All button</td>
<td>Click to display the selected member and any children under the selected member in an expanded format (default)</td>
</tr>
<tr>
<td>Collapse All button</td>
<td>Click to display the selected member and any children under the selected member in a collapsed format</td>
</tr>
<tr>
<td>&lt;&lt;FirstDiff button</td>
<td>Click to jump to the first model element with a difference</td>
</tr>
<tr>
<td>Element</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>&lt;PrevDiff button</td>
<td>Click to display the difference immediately previous to the current difference</td>
</tr>
<tr>
<td>NextDiff&gt; button</td>
<td>Click to display the next difference after the current difference</td>
</tr>
<tr>
<td>LastDiff&gt;&gt; button</td>
<td>Click to jump to the last model element with a difference</td>
</tr>
<tr>
<td>View All button</td>
<td>Click to display all model elements, not just the elements with differences</td>
</tr>
<tr>
<td>Show Diff Only button</td>
<td>Click to display only the model elements with differences (default)</td>
</tr>
<tr>
<td>Note:</td>
<td>For contextual purposes, Show Diff Only also displays the members immediately previous to and immediately after the member with a difference.</td>
</tr>
<tr>
<td>View button</td>
<td>Click to display the member property differences for a selected element</td>
</tr>
<tr>
<td>&lt;--</td>
<td>A red arrow indicates a deleted element in the Application Version of a model</td>
</tr>
<tr>
<td>--&gt;</td>
<td>A green arrow indicates an inserted element in the Application Version of a model</td>
</tr>
<tr>
<td></td>
<td>Click to jump to the first page of the model</td>
</tr>
<tr>
<td></td>
<td>Click to display the previous page of the model</td>
</tr>
<tr>
<td>Page drop-down list box</td>
<td>Select a page to display in the Taskflow Listing area.</td>
</tr>
<tr>
<td></td>
<td>Click to display in the Taskflow Listing area the page you selected in the Page drop-down list box.</td>
</tr>
<tr>
<td></td>
<td>Click to display the next page of the model</td>
</tr>
<tr>
<td></td>
<td>Click to jump to the last page of the model</td>
</tr>
<tr>
<td>Rows</td>
<td>The number of rows displayed on each page (default is 50)</td>
</tr>
</tbody>
</table>

**Viewing Model Content**

Shared Services provides a viewer that enables you to view the content of models directly in Shared Services. You can use the viewer only with dimensional mode. You need Read permission to view the content of a model.

The viewer enables you to:

- View all members for a model, including application system members
- Rename a member

When renaming a member:

1. You cannot create a duplicate member name.
2. You cannot rename an application system member.
To view dimension members:

1. From Financial Management, select Administration > Manage Models.
2. If it is not already selected, select Browse.
3. Select a model and click View.
   The dimension viewer shows the members of the selected model, including any application system members.
4. Click Validate to perform a simple validation check.
   The validation check verifies these facts and lists exceptions:
   - That you have not created names that are too long (for example, 20 characters for Financial Management, 80 characters for Oracle’s Hyperion® Planning – System 9)
   - That you have not created duplicate names

   **Note:**
   Shared Services does not validate that Aliases or UDAs are unique.

### Application System Members

Application system members store critical system information such as currency rates and ownership information. Each product has certain application system members that, when exported to Shared Services, are displayed in the model hierarchy.

You can view the details and properties of an application system member, however you cannot delete, edit, add children to, or rename an application system member in Shared Services.

Application system members are filtered out of the hierarchy if they are not relevant to the product being viewed. The compare operation filters out any application system members that are not valid for your product. For example, if you are viewing Financial Management models, Shared Services filters out any application system members that are not valid for Financial Management.

You can export models that contain application system members. While performing these sync operations:

- Export with Overwrite replaces the Shared Services model with the application model, including any application system members.

- Export with Merge merges the application model content with the content in Shared Services. Notice:
  - If an application system member exists only in Shared Services, export with merge does not delete this member.
  - If an application system member exists both in Shared Services and in the product, export with merge will merge the properties as usual and take the system member-specific attributes from the product side of the model.
  - All other export with merge scenarios will behave exactly the same way for system members as they do for normal members.
Renaming Models

Renaming a model changes the name only in Shared Services. The internal representation of the name does not change.

You need Write access to a model to rename it.

➤ To rename a model:
1. From Financial Management, select Administration > Manage Models.
2. If it is not already selected, select Browse.
3. Select a model and click Rename.
4. Type a name in New Name.
5. Click an option:
   ● Rename to save the new name
   ● Cancel to cancel the name change

Deleted Models

You can delete a model if you have Write access to it.

➤ To delete a model:
1. From Financial Management, select Administration > Manage Models.
2. If it is not already selected, select Browse.
3. Select a model and click Delete.
4. Click OK to confirm deletion.

Tracking Model History

Shared Services maintains a version history for each model in the product, if versioning is enabled for the model.

➤ To view the version history of a model in Shared Services:
1. From Financial Management, select Administration > Manage Models.
2. If it is not already selected, select Browse.
3. Select a model and click History.
   
   Shared Services displays a list of model versions, including the name of the person who updated the version, the update date, and comments for each model.
4. From the version list, perform a task:
   a. View the contents of any model.
i. Click a version.
ii. Click View.

b. Compare two model versions.
i. Select two versions.
ii. Click Compare.

The contents of the two model versions are shown line-by-line in a side-by-side format.

c. View the properties of a model.
i. Click a version.
ii. Click Properties.

Managing Permissions to Models

Shared Services enables you to manage access permissions to models in applications independent of any application. You assign permissions on a model-by-model basis to individual users or to groups of users. You can also assign permissions at the application level.

When applications are created in the Shared Services repository, all permissions are denied for all users, except the user who created the application. The creator of the application or the administrator must assign permissions for the new application.

To access specific models in Shared Services, users must be assigned access rights individually or inherit access rights by being part of a group that is assigned access rights. If an individual user is assigned to a group and the access rights of the individual user conflict with those of the group, the rights of the individual user take precedence.

To give users access to models other than their own, an administrator must add the users and assign their permissions.

Permissions

Model management provides these permission types:

- **Read**—The ability to view the contents of a model.
- **Write**—The ability to export a model.

Write access does not automatically include Read permission. You must assign Read permission explicitly, in addition to Write permission, if you want a user to have these permissions.

- **Manage**—The ability to create new users and change permissions for users.

Manage access does not automatically include Read and Write permissions. You must assign Read and Write permissions explicitly, in addition to Manage permission, if you want a user to have all these permissions.

You can assign these access permissions to models.
Table 94  Access Permissions

<table>
<thead>
<tr>
<th>Access Permission</th>
<th>Action</th>
<th>Read</th>
<th>Write</th>
<th>Manage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sync</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Export</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>View</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Filter</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Compare</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>History</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Set Properties</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Assign Access</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Assign Permissions</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

You can apply permissions to groups and to individual users. Users are automatically granted the permissions of the groups to which they belong. You can, however, explicitly add or deny permissions to a user to override group permissions.

For each type of access permission (Read, Write, and Manage), you must apply one of the following actions:

- **Grant**—Explicitly grant the permission to the user or group.
  
  Granting permissions to a member of a group overrides permissions inherited from the group. For example, if a group is denied a permission, you can explicitly grant the permission to a member of the group.

- **Deny**—Explicitly deny the permission to the user or group.
  
  Denying permissions to a member of a group overrides permissions inherited from the group. For example, if a group is granted a permission, you can explicitly deny the permission to a member of the group.

- **None**—Do not apply the permission to the user or group.
  
  Not applying a permission is different from denying a permission. Not applying a permission does not override permissions inherited from a group. Specifying None for particular permissions for individual users enables you to apply permissions on a group basis.

**Note:**

If a user belongs to groups with mutually exclusive permissions to the same model, permissions that are assigned override permissions that are denied. For example, if a user belongs to a group that denies Read access to a particular model and belongs to another...
group that assigns Read access to the model, the user in fact is granted Read access to the model.

Assigning Permissions to Models

You assign permissions on individual models in applications. You assign permissions on the models to individual users or to groups of users. You must have Manage permission for a model to assign permissions to it.

To assign permissions to models:

1. From Financial Management, select Administration > Manage Models.
2. If it is not already selected, select Browse.
3. Select a model and click Access.
   You can view the permissions that are assigned to users and groups for the selected model in the Model Access window.
4. To add users or groups, click Add.
   The Available Users/Groups text box lists users who are authenticated as Shared Services users. If a user that you want is not on the list, contact the Shared Services administrator, who can use the Shared Services Configuration Console to add authenticated users.
5. In Available Users/Groups, select users or groups to assign to this model (press Ctrl to select multiple users). Click Add to move the selected users and groups to Selected Users/Groups, or click Add All to move all users and groups to Selected Users/Groups.
   Note:
   Group names are preceded by an asterisk (*).
6. Assign permissions to the selected users and groups by selecting one of the Grant, Deny, or None options for the Read, Write, and Manage permissions.
   Note:
   Assigning (or denying) a permission does not implicitly assign (or deny) any other permissions; that is, assigning Write permission does not implicitly assign Read permission, and assigning Manage permission does not implicitly assign Read and Write permissions. Likewise, denying Read permission does not implicitly deny Write and Manage permissions, and denying Write permission does not implicitly deny Manage permission. You must explicitly assign all permissions that you want a user to have.
7. Click Add to assign the new permissions.

Deleting Permissions to Models

You can delete all permissions for users and groups to individual models. You must have Manage permission for a model to delete access to it.
To delete access to a model:

1. From Financial Management, select Administration > Manage Models.
2. If it is not already selected, select Browse.
3. Select a model name and click Access.
   You can view the permissions that are assigned to users and groups for the selected model.
4. Select the check box next to one or more users or groups and click Delete.

Note:
When you click Delete, the permissions are immediately removed without displaying a warning message being displayed.

Viewing and Setting Model Properties

Shared Services provides property data for each model in the product. You can view all model properties and set selected properties.

Shared Services displays these model properties:

- **Creator**—Name of the user who created the model
- **Updated By**—Name of the person who updated the model
  
  If there have been no updates, the name of the creator is listed and the Updated Date is the same as the Created date.

- **Create Date**—The date on which the model was created in (exported to) Shared Services

- **Updated Date**—The date on which the model was last updated in Oracle's Hyperion® Shared Services

- **Versioning**—Whether versioning is enabled
  
  If versioning is not enabled, you can enable it by changing this setting. Once versioning is enabled, however, you cannot disable it.

- **Lock Status**—Whether the model is locked or unlocked
  
  You can change this setting to lock the model for your exclusive use or to unlock the model to allow other users to work with it. Models are locked for only 24 hours. After 24 hours, the model is automatically unlocked.

You need Read access to view model properties and Write access to change model properties.

To view or change model properties:

1. From Oracle's Hyperion® Financial Management – System 9, select Administration > Manage Models.
2. If it is not already selected, select Browse.
3. Select a model and click Properties.
   You can view the model properties.
4 You can make these changes to model properties:

- If versioning is not enabled, enable it by clicking Enable next to Versioning. After versioning is enabled, model management maintains a version history for the model. You cannot disable versioning for a model after you enable it.

- Lock or unlock the model by clicking Lock or Unlock next to Lock Status.

- If the Dimension Type value is None, select a new dimension type in the drop-down next to Change To. After you select a new dimension type, click Change To and accept the resulting confirmation to invoke the change.

5 Click Close to return to the previous page and save changes.
You can use submission phases for Process Management, which enable you to subdivide a process unit into different phases and apply process management to those phases. 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.

You can use a maximum of nine submission phases in the review process. Submission phases can vary by period and scenario. For example, in the Actual scenario, you might submit Balance Sheet and Profit/Loss accounts for review in the first submission phase, and supplemental data in the second submission phase. In the Budget scenario, you might submit Intercompany data in the first submission phase, Balance Sheet and Profit/Loss accounts in the second phase, and supplemental data in the third phase.

You define and set up submission phases in metadata files. See the Hyperion Enterprise Performance Management Architect Administrator’s Guide.

### Defining Submission Phases by Scenario and Period

Your review process requirements may vary from period to 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 supplemental data and Balance Sheet and Profit/Loss data.

Phase definition can also differ 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.

To set up submission phases, administrators must complete these procedures:
● In the metadata file, set the application and dimension metadata attributes to use submission phases. Set the UserSubmissionPhase application attribute to Y, and set SupportSubmissionPhaseForAccount, SupportSubmissionPhaseForCustom, or SupportSubmissionPhaseForICP attributes as required. For example, if the application only needs submission by accounts and not for Custom 1 to 4 or Intercompany (ICP) dimensions, you can select the SupportSubmissionPhaseForAccount attribute. At least one dimension must be enabled.

● Define submission groups and assign submission groups to dimensional members (accounts, Custom 1 to 4 dimensions, ICP members)

● Assign submission groups to submission phases

● Assign validation accounts to submission groups. Validation accounts are used to ensure that the value equals zero before a process unit can be promoted to the next review level. Assign Submission Group 0 to validation accounts that are not subject to process management.

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 Assignments by Scenario and Period

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.

If a group is not specified, process management is not applied.

The following examples show sample submission groups and their assignments to submission phases.

<table>
<thead>
<tr>
<th>Accounts</th>
<th>Submission Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>HistData</td>
<td>0</td>
</tr>
<tr>
<td>Cash</td>
<td>1</td>
</tr>
<tr>
<td>Invest</td>
<td>1</td>
</tr>
<tr>
<td>ICR</td>
<td>2</td>
</tr>
<tr>
<td>ICP</td>
<td>2</td>
</tr>
<tr>
<td>Liability</td>
<td>3</td>
</tr>
<tr>
<td>Equity</td>
<td>3</td>
</tr>
<tr>
<td>Revenue</td>
<td>4</td>
</tr>
<tr>
<td>Expense</td>
<td>4</td>
</tr>
<tr>
<td>SuppData1</td>
<td>5</td>
</tr>
<tr>
<td>SuppData2</td>
<td>5</td>
</tr>
<tr>
<td>Headcount</td>
<td>6</td>
</tr>
<tr>
<td>MiscData</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C1 (Product)</th>
<th>Submission Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>{None}</td>
<td>1</td>
</tr>
<tr>
<td>Golf Balls</td>
<td>7</td>
</tr>
<tr>
<td>Tennis Balls</td>
<td>8</td>
</tr>
<tr>
<td>Soccer Balls</td>
<td>9</td>
</tr>
</tbody>
</table>

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. If you leave the submission group blank, it defaults to Submission Group 1. In this example, the HistData account has a submission group assignment of 0, which means that the account does not require review process.
The following example shows submission group assignments by period for the Actual scenario.

**Table 95  Example: Submission Group Assignment to Submission Phases**

<table>
<thead>
<tr>
<th>Period</th>
<th>Submission Phase 1</th>
<th>Submission Phase 2</th>
<th>Submission Phase 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>January (Single Phase)</td>
<td>1, 2, 3, 4</td>
<td>N/A (No supplemental or product data required)</td>
<td>N/A (No supplemental or product data required)</td>
</tr>
<tr>
<td>February</td>
<td>1, 2, 3, 4</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>March (Multiple Phases)</td>
<td>2</td>
<td>1, 3, 4, 7, 8, 9</td>
<td>5, 6</td>
</tr>
<tr>
<td>April</td>
<td>1, 2, 3, 4</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>May</td>
<td>1, 2, 3, 4</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>June (Multiple Phases)</td>
<td>2</td>
<td>1, 3, 4, 7, 8, 9</td>
<td>5, 6</td>
</tr>
<tr>
<td>July</td>
<td>1, 2, 3, 4</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>August</td>
<td>1, 2, 3, 4</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>September (Multiple Phases)</td>
<td>2</td>
<td>1, 3, 4, 7, 8, 9</td>
<td>5, 6</td>
</tr>
<tr>
<td>October</td>
<td>1, 2, 3, 4</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>November</td>
<td>1, 2, 3, 4</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>December (Multiple Phases)</td>
<td>2</td>
<td>1, 3, 4, 7, 8, 9</td>
<td>5, 6</td>
</tr>
</tbody>
</table>

**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 ICRrec 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 display and manage submission phases using the Manage Submission Phases task. This task only displays if you enable the UseSubmissionPhase application setting in the metadata file. You must also be assigned to the administrator or Review Supervisor security role.

In the Manage Submission Phases window, you can specify the submission phases to display, from 1 to 9. You can change the setting as needed to display different submission phases.

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. You can use the asterisk (*) keyword to indicate all groups. The default for Submission Phase 1 is the asterisk (*) 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 in Phase 1 and 3 but not in 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. In the Browser view, select Tasks > Data Tasks > Manage Submission Phases.
2. Select the phases to display, and click OK.
3. To change the scenario, click on it, select a scenario, and click OK.
4. In each phase column, enter the groups for that submission phase and click 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, use an asterisk (*).
5. Click Submit Data to save the data, or click Reset Data 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:

<table>
<thead>
<tr>
<th>Period</th>
<th>Submission Phase 1</th>
<th>Submission Phase 2</th>
<th>Submission Phase 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>January</td>
<td>1,2,3,4</td>
<td>5,7</td>
<td>8,9</td>
</tr>
<tr>
<td>February</td>
<td>1,2,3,4</td>
<td>5–8</td>
<td>N/A</td>
</tr>
<tr>
<td>March</td>
<td>2</td>
<td>1,3,4</td>
<td>5,6</td>
</tr>
<tr>
<td>April</td>
<td>1,2,3,4</td>
<td>5,6</td>
<td>8</td>
</tr>
<tr>
<td>May</td>
<td>1,2,3,4</td>
<td>5–8</td>
<td>N/A</td>
</tr>
<tr>
<td>June</td>
<td>2</td>
<td>1,3,4</td>
<td>5,6</td>
</tr>
<tr>
<td>July</td>
<td>1,2,3,4</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>August</td>
<td>1,2,3,4</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>September</td>
<td>2</td>
<td>1,3,4</td>
<td>5,6</td>
</tr>
<tr>
<td>October</td>
<td>1,2,3,4</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>November</td>
<td>1,2,3,4</td>
<td>NA</td>
<td>N/A</td>
</tr>
<tr>
<td>December</td>
<td>2</td>
<td>1,3,4</td>
<td>5,6</td>
</tr>
</tbody>
</table>

If you select the option to display unassigned groups, for January in the previous example, these groups are shown as unassigned:

<table>
<thead>
<tr>
<th>Period</th>
<th>Unassigned Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>January</td>
<td>6,10</td>
</tr>
</tbody>
</table>

To view unassigned submission groups:

1. In the Browser View, select Tasks > Data Tasks > Manage Submission Phases.
2. Select the cell for the phase, right-click and select Show Unassigned Groups.
3. From the popup list, review the list of unassigned groups if applicable, and then click OK.
account  A dimension that represents an accounting container that identifies the location and primary nature of the data.

account type  The definition of how an account's value flows over time, and its sign behavior. Account type options can include expense, income, asset, liability, and equity. Expense examples: payroll expenses, salaries, office supplies, legal expenses, and rent. Revenue examples: sales, interest income, and other income. Asset examples: cash, accounts receivable, fixed assets, and accumulated depreciation. Liability examples: accounts payable, accrued expenses, and long-term debt. Equity examples: common stock, preferred stock, additional paid-in-capital, and retained earnings.

activity-level authorization  Defines user access to application components and the types of activities they can perform on the application component. Activity-level authorization controls whether a given user may perform a certain action in an application and is independent of the data that will be operated on by the action. Data access is controlled by data-level authorization.

alias  An alternative name. For example, for a more easily identifiable column descriptor you can display the alias instead of the member name.

ancestor  A branch member that has members below it. For example, in a dimension that includes years, quarters, and months, the members Qtr2 and 2007 are ancestors of the member April.

application  (1) A software program designed to run a specific task or group of tasks such as a spreadsheet program or database management system. (2) A related set of dimensions and dimension members that are used to meet a specific set of analytical and/or reporting requirements.

application currency  The default reporting currency for the application.

asset account  An account type that stores values that represent a company's assets.

attribute  Characteristics of a dimension member. For example, an Employee Number dimension member may have attributes of Name, Age, or Address or a product dimension can have several attributes, such as a size and flavor.

authentication  Verification of identity as a security measure. Authentication is typically based on a user name and password. Passwords and digital signatures are forms of authentication.

auto-reversing journal  A journal for entering adjustments that you want to reverse in the next period.

balance account  An account type that stores unsigned values that relate to a particular point in time.

balanced journal  A journal in which the total debits equal the total credits.

base currency  The currency in which daily business transactions are performed.

base entity  An entity at the bottom of the organization structure that does not own other entities.

business rules  Logical expressions or formulas that are created within an application to produce a desired set of resulting values.

CALC status  A consolidation status that indicates that some values or formula calculations have changed. You must reconsolidate to get the correct values for the affected entity.

calendar  User-defined time periods and their relationship to each other. Q1, Q2, Q3, and Q4 comprise a calendar or fiscal year.
cell  (1) The data value at the intersection of dimensions in a multidimensional database; the intersection of a row and a column in a worksheet. (2) A logical group of nodes belonging to one administrative domain.

CHANGED status  A consolidation status that indicates data has changed for an entity.

child  A member with a parent above it in the database outline.

column  A vertical display of information in a grid or table. A column can contain data from one field, derived data from a calculation, or textual information.

consolidation  The process of gathering data from dependent entities and aggregating the data to parent entities. For example, if the dimension Year consists of the members Qtr1, Qtr2, Qtr3, and Qtr4, its consolidation is Year.

consolidation rule  Identifies the rule that is executed during the consolidation of the node of the hierarchy. This rule can contain customer specific formulas appropriate for the correct consolidation of parent balances. Elimination processing can be controlled within these rules.

contribution  The value added to a parent from a child entity. Each child has a contribution to its parent.

conversion rate  See exchange rate.

currency conversion  A process that converts currency values in a database from one currency into another currency. For example, to convert one U.S. dollar into the European euro, the exchange rate (for example, 0.923702) is multiplied with the dollar (1 * 0.923702). After conversion, the European euro amount is .92.

custom dimension  A dimension created and defined by users. For example, channel, product, department, project, or region could be custom dimensions.

data form  A grid display that enables users to enter data into the database from an interface such as a Web browser, and to view and analyze data or related text. Certain dimension member values are fixed, giving users a specific view into the data.

data lock  A user-controlled mechanism that prevents changes to data according to specified criteria, such as period or scenario.

dependent entity  An entity that is owned by another entity in the organization.

destination currency  The currency to which the balances are converted. You enter exchange rates and convert from the source currency to the destination currency. For example, when you convert from EUR to USD, the destination currency is USD.

dimension  A data category used to organize business data for retrieval and preservation of values. Each dimension usually contains a hierarchy of related members grouped within it. For example, a Year dimension often includes members for each time period, such as quarters and months.

direct rate  A currency rate that you enter directly in the exchange rate table. The direct rate is used for currency conversion. For example, assume you want to convert balances from JPY to USD. In the exchange rate table, you enter a rate for the period/scenario where the source currency is JPY and the destination currency is USD.

drill-down  Navigation through the query result set using the organization of the dimensional hierarchy. Drilling down moves the user perspective from general aggregated data to more detailed data. For example, drilling down can reveal the hierarchical relationships between year and quarters or between quarters and months.

dynamic member list  A named member set that is a system-created list based on predefined criteria. The dynamic member list is automatically refreshed whenever the list is referenced in the application. As dimension members are added and deleted, the dynamic member list automatically reflects the changes by reapplying the user-defined criteria.

dynamic view account  An account type indicating that the account value is calculated dynamically from the data that you are viewing.

elimination  The process that occurs when certain transactions are zeroed out or “eliminated” because they were generated between entities within the same organization.

entity  A dimension representing organizational units. Examples include divisions, subsidiaries, plants, regions, products, or other financial reporting units.
**exchange rate**  A numeric value used to convert one currency to another. For example, to convert 1 USD into EUR, the exchange rate of 0.8936 is multiplied with the U.S. dollar. The European euro equivalent of $1 is 0.8936.

**exchange rate type**  An identifier associated with an exchange rate. Different rate types are used because there may be multiple rates for a period and year. Users traditionally define a rate at period end for the average rate of the period and also a rate for the end of the period. Additional rate types are historical rates, budget rates, forecast rates, and so on. All these exchange rate types apply to the same point in time.

**expense account**  An account type that stores periodic and year-to-date values that decrease net worth if the value is positive.

**Extensible Markup Language (XML)**  A language comprised of a set of tags used to assign attributes to data that can be interpreted between applications based on the schema used.

**external authentication**  Logging on to Oracle’s Hyperion applications by means of user information stored outside the application, typically in a corporate user directory such as MSAD or NTLM.

**file delimiter**  One or more characters, such as commas or tabs, that separate fields in a data source.

**flow account**  An unsigned account type that stores periodic and year-to-date values.

**function**  A predefined routine that returns a value, a range of values, a Boolean value, or one or more database members.

**generation**  A layer in a hierarchical tree structure that defines member relationships in a database. Generations are ordered incrementally from the top member of the dimension (generation 1) down to the child members.

**holding company**  An entity that is part of a legal group of entities, and has either a direct or indirect investment in all the other entities within the legal group.

**IMPACTED status**  A consolidation status indicating that the data that rolls up to this parent entity has changed.

**INACTIVE status**  A consolidation status indicating this entity is not active for the current period.

**intercompany elimination**  See elimination.

**intercompany matching**  The process of comparing balances for pairs of intercompany accounts within an application. Intercompany receivables are typically matched or compared to intercompany payables. The system uses these matching accounts to eliminate any intercompany transactions from your organization’s consolidated totals.

**intercompany matching report**  A report that compares the balances of a group of intercompany accounts and indicates if the accounts are in balance or out of balance for any branch of an organization.

**journal entry (JE)**  A set of debit/credit adjustments to account balances for a scenario and period.

**liability account**  An account type that stores “point in time” balances that represent a company’s liabilities. Examples of liability accounts include accrued expenses, accounts payable, and long term debt.

**locked**  A user-invoked process that prevents the data from being modified by any process, user, or other means.

**LOCKED status**  A consolidation status indicating that an entity contains data that is locked for the specified period.

**member list**  A named group, system- or user-defined, that references members, functions, or other member lists within a dimension.

**merge**  A data load option that clears existing values from the accounts specified in the data load file and then replaces them with the values in the data load file. Unlike the Replace option, the Merge option clears only the values from the accounts specified in the load file.

**metadata**  A set of data that defines and describes the properties and attributes of the data stored in a database or used by an application. Examples of metadata are dimension names, member names, properties, time periods, and security.

**NO DATA status**  A consolidation status indicating that this entity contains no data for the specified period and account.
ODBC  Open Database Connectivity. A database access method used from any application without regard to how the database management system (DBMS) processes the information.

OK status  A consolidation status indicating that an entity has already been consolidated, and that data has not changed below it in the organization structure.

Open Database Connectivity (ODBC)  Standardized application programming interface (API) technology that allows applications to access multiple third-party databases.

organization  A hierarchy of entities that defines each entity and their relationship to others in the hierarchy.

P&L accounts (P&L)  Profit and loss accounts. Refers to a typical grouping of expense and income accounts that comprise a company's income statement.

parent adjustments  The journal entries that are posted to a child in relation to its parent.

parents  The entities that contain one or more dependent entities that report directly to them. Because parents are both entities and associated with at least one node, they have entity, node, and parent information associated with them.

percent consolidation  The portion of a child's values that is consolidated to its parent.

percent control  Identifies the extent to which an entity is controlled within the context of its group.

percent ownership  Identifies the extent to which an entity is owned by its parent.

periodic value method (PVA)  A process of currency conversion that applies the periodic exchange rate values over time to derive converted results.

plug account  An account in which the system stores any out of balance differences between intercompany account pairs during the elimination process.

promote  The action to move a data unit to the next review level, allowing a user having the appropriate access to review the data. For example, an analyst may promote the data unit to the next level for his supervisor's review.

PVA  See periodic value method (PVA).

regular journal  A feature used to enter one-time adjustments for one period only. Regular journals can be balanced, balanced by entity, or unbalanced.

relational database  A type of database that stores data in the form of related two-dimensional tables. Contrast with periodic value method (PVA).

replace  A data load option that clears the existing values from all accounts for the periods specified in the data load file, and then loads the values from the data load file. If an account is not specified in the load file, its values for the specified periods are cleared during the load.

review level  A Process Management review status indicator that represents the process unit level. Review levels include the following: Not Started, First Pass, Review Level 1, Review Level 2 ... Review Level 10, Submitted, Approved, and Published.

rules  User-defined formulas.

scenario  A dimension that specifies a data classification. Examples include Actuals, Budget, Forecast1, and Forecast2.

source currency  The currency from which the balances originate and are converted. You enter exchange rates and convert from the source currency to the destination currency. For example, when you convert from EUR to USD, the source currency is EUR.

standard journal template  A journal function used to post adjustments that have common adjustment information for each period. Instead of creating a new regular journal every month, you can create a standard template that contains the common account IDs, entity IDs, or amounts. You can then use the template as the basis for many regular journals that contain similar adjustment information.

system extract  A feature that allows you to transfer data from an application's metadata into an ASCII file.

translation  See currency conversion.
**triangulation**  A means of converting balances from one currency to another via a third common currency. In Europe, this is the euro for member countries. For example, to convert from French franc to Italian lira, the common currency is defined as European euro. Therefore, in order to convert balances from French franc to Italian lira, balances are converted from French franc to European euro and from European euro to Italian lira.

**triangulation currency**  A currency through which exchange rates can be derived. For example, if you set up the Euro/Dollar exchange rate and the Euro/Yen rate, the Dollar/Yen rate can be derived by using Euro as the triangulation currency.

**unbalanced journal**  A journal in which the total debits do not equal the total credits.

**user-defined member list**  A named, static set of members within a dimension defined by the user. As dimension members are added to an application, member lists remain constant and will not reflect any changes made to the application outline.

**value dimension**  A dimension that is used to define the input value, translated value, and consolidation detail.

**view**  The representation of either a year-to-date or periodic display of data.

**XML**  See *Extensible Markup Language (XML)*.
Index

A
ABSEExp function, 242
account attributes, 75
Account dimension, 18
account expressions, rules, 218
accounts example, 100
AccountType attribute, 75
behavior, 78
AccountType function, 243
AccountTypeID function, 244
active account, 93
AddEntityToList function, 245
AddEntityToListUsingIDs function, 246
AddMember keyword, Web scripts, 150
AddMemberToList function, 247
alerting
intercompany transactions, 62
process management, 61
Alloc function, 248
AllowAdjFromChildren attribute, 82
AllowAdjFromChildren function, 249
AllowAdjS attribute, 82
AllowAdjS function, 250
APP or XML files, creating, 97
APP to XML, 107
application profiles
creating, 25
defining calendars for, 26
saving, 32
specifying frequencies for, 27
specifying languages for, 25
specifying periods for, 28
application profiles (PER), creating, 25
application security files (SEC)
extracting in Web, 69
extracting in Windows, 68
loading in Web, 64
loading in Windows, 63
application settings attributes, 85
application settings example, 99
applications
closing, 36
creating, 32
creating in Web, 34
creating in Windows, 33
deleting, 36
opening, 35
asset account behavior, 78
assigning access to security classes, 59
assigning users and groups to roles, 59
attributes
AccountType, 75
AllowAdjFromChildren, 82
AllowAdjS, 82
ConsolidateYTD, 83
ConsolidationRules, 86
ConsolMethod, 89
Control, 89
Currency, 91
CustomTopMember, 76
DefaultCurrency, 86
DefaultFreq, 83
DefaultParent
accounts, 76
custom, 80
entities, 82
scenarios, 83
DefaultRateForBalanceAccounts, 86
DefaultRateForFlowAccounts, 86
DefaultValueForActive, 86
DefaultView, 83
DefCurrency, 82
DefFreqForICTrans, 84
Description
accounts, 76
consolidation methods, 89
currencies, 91
custom, 80
entities, 82
scenarios, 84
EnableCustomAggr, 76
EnableDataAudit
accounts, 76
scenarios, 84
EnableMetadataSecurityFiltering, 86
HoldingCompany, 82
ICPEntitiesAggregationWeight, 86
ICPTopMember
accounts, 76
IsCalculated
accounts, 76
custom, 80
IsConsolidated, 76
IsHoldingMethod, 89
IsICP
accounts, 76
entities, 82
MaximumReviewLevel, 84
Member
accounts, 77
custom, 81
entities, 82
scenarios, 84
NodeSecurity, 87
NumDecimalPlaces, 77
OrgByPeriodApplication, 87
PercentConsol, 89
PlugAcct, 77
Scale, 91
SecurityAsPartner, 83
SecurityClass
accounts, 77
custom, 81
entities, 83
scenarios, 84
Submission Group
accounts, 77, 81
SupportsProcessManagement, 84
SwitchSignForFlow, 81
SwitchTypeForFlow, 81
ToPercentControl, 90
ToPercentControlComp, 90
TranslationOperator, 92
UsedByCalcRoutine, 90
UsePVAForBalanceAccounts, 87
UsePVAForFlowAccounts, 87
UserDefined1...3
accounts, 78
custom, 82
entities, 83
scenarios, 85
UseSecurityForAccounts, 87
UseSecurityForCustom, 88
UseSecurityForEntities, 88
UseSecurityForICP, 88
UseSecurityForScenarios, 88
UsesLineItems
accounts, 78
scenarios, 85
UseSubmissionPhase, 88
ValidationAccount, 88
XBRLTags, 78
ZeroViewForAdj, 85
ZeroViewForNonadj, 85
auditing data, 48
auditing tasks, 45

B
BackgroundPOV keyword, Web scripts, 151
balance account behavior, 78
balance recurring account behavior, 78
Blank keyword, Web scripts, 152

C
Calc1 keyword, Web scripts, 153
CalcByRow keyword, Web scripts, 153
CalcStatus function, 252
CalculateExchangeRate function, 250
CalculateRate function, 251
Cell_Link keyword, Web scripts, 154
CellText keyword, Web scripts, 154
Clear function, 253
closing applications, 36
Cn keyword
Web scripts, 152
Con function, 254
Consoll, Consol2, Consol3 function, 255
Consoll...3 account, 93
ConsolidateYTD attribute, 83
ConsolidateYTD function, 256
consolidation accounts
  active account, 93
  Consol1...3 account, 93
  DOWN account, 93
  method account, 93
  PCON account, 93
  PCTRL account, 93
  POWN account, 93
consolidation methods, 89
consolidation rules, execution order, 220
ConsolidationRules attribute, 86
ConsolMethod attribute, 89
ContainsCellText function, 256
Control attribute, 89
copying periods, 31
CreateInputCache function, 257
creating
  application profiles, 25
  application profiles (PER), 25
  applications, 32
data form scripts (WDF), 147
data forms in Form Builder, 177
  folders for data forms, 182
in Web applications, 34
in Windows applications, 33
  journal files (JLF), 137
  member list files (LST), 127
  metadata files (APP or XML), 97
rules files (RLE), 229
  rules, Rules Editor, 227
  star schema template, 193
CSV file
  data audit log, 49
  task audit log, 48
CUR keyword
  Web scripts, 175
Currencies attributes, 91
currencies example, 99
Currency attribute, 91
Currency function, 258
custom attributes, 80
Custom dimension, 18
custom dimension example, 101
Custom1Top, Custom2Top, Custom3Top, and
  Custom4Top functions, 258
CustomHeader keyword
Web scripts, 155
  CustomHeaderStyle keyword
  Web scripts, 156
  CustomTopMember attribute, 76

D
data audit, 48
Data Explorer reports, 196
data form script syntax, 147
data form scripts (WDF)
  creating, 147
  extracting, 184
  loading, 183
data forms
  creating folders, 182
deleing, 184
Decimal function, 259
default translation, 220, 221
DefaultCurrency attribute, 86
DefaultFreq attribute, 83
DefaultFreq function, 260
DefaultParent attribute
  accounts, 76
custom, 80
entities, 82
scenarios, 83
DefaultRateForBalanceAccounts attribute, 86
DefaultRateForFlowAccounts attribute, 86
DefaultTranslate function, 261
DefaultValueForActive attribute, 86
DefaultView attribute, 83
DefaultView function, 261
DefCurrency attribute, 82
DefCurrency function, 262
DefFreqForICTrans attribute, 84
defining
  account attributes, 75
  application settings, 85
calendar in applications, 26
consolidation methods, 89
Currencies attributes, 91
custom attributes, 80
tenity attributes, 82
scenario attributes, 83
deleting
  applications, 36
data forms, 184
periods, 31
star schema, 193
star schema template, 194
delimiter characters, 18
Description attribute
accounts, 76
consolidation methods, 89
currencies, 91
custom, 80
entities, 82
scenarios, 84
description requirements, 18
dimensions overview, 16
disabling connections for users on system, 44
DOWN account, 93
DOwn function, 263
Dynamic function, 264
dynamic member lists, 130
DynamicPOV keyword, Web scripts, 156

E
E-mail alerting
intercompany transactions, 62
process management, 61
editing data forms in Form Builder, 181
editing forms, 181
EnableCustomAggr attribute, 76
EnableDataAudit attribute
accounts, 76
scenarios, 84
EnableMetadataSecurityFiltering attribute, 86
EnableProcessManagement function, 265
enabling connections for users on system, 44
entities example, 101
entity attributes, 82
Entity dimension, 17
Exp function, 266
expense account behavior, 78
exporting data to new star schema, 191
Expression Builder
using the Function Wizard, 228
extracting
data form scripts (WDF), 184
in Web
application security files (SEC), 69
journal files (JLF), 144
member list files (LST), 134
metadata files (XML and APP), 124
rules files (RLE), 232
in Windows
application security files (SEC), 68
journal files (JLF), 143
member list files (LST), 133
metadata files (XML and APP), 123
rules files (RLE), 231

F
File Properties
creating metadata reports, 116
Financial Management
architecture, 16
dimensions, 16
FIRST keyword
Web scripts, 175
flow account behavior, 78
Form Builder, 177
editing forms, 181
FormInputBoxLength keyword, Web scripts, 156
FormNumDecimals keyword
Web scripts, 157
FormRowHeight keyword, Web scripts, 157
FormScale keyword, Web scripts, 157
frequencies, 27
Function Wizard, using in Expression Builder, 228
functions
ABSExp, 242
AccountType, 243
AccountTypeID, 244
AddEntityToList, 245
AddEntityToListUsingIDs, 246
AddMemberToList, 247
Alloc, 248
AllowAdjFromChildren, 249
AllowAdj, 250
CalcStatus, 252
CalculateExchangeRate, 250
CalculateRate, 251
Clear, 253
Con, 254
Consol1, Consol2, Consol3, 255
ConsolidateYTD, 256
ContainsCellText, 256
CreateInputCache, 257
Currency, 258
GetCell, 272
GetCellNoData, 273
GetCellRealData, 274
GetCellType, 275
GetItem, 275
GetItemIDs2, 276
GetNumItems, 277
GetNumLID, 277
GetRate, 278
GetSubmissionGroup, 279
GetSubmissionPhase, 279
Holding, 280
ICPTopMember, 280
ICPWeight, 281
IDFromMember, 282
ImpactStatus, 283
Input, 284
IsAlmostEqual, 284
IsBase, 285
IsCalculated, 286
IsChild, 287
IsConsolidated, 289
IsDescendant, 289
IsFirst, 291
IsICP, 292
IsLast, 293
IsTransCur, 293
IsTransCurAdj, 294
IsValidDest, 294
IsZero, 295
List, 296
Member, 297
MemberFromID, 298
Method, 299
NoInput, 299
NoRound, 300
NumBase, 301
Number, 302
NumChild, 303
NumDescendant, 304
OpenDataUnit, 306
overview, 236
PCon, 307
PlugAcct, 307
POwn, 308
PVAForBalance, 309
PVAForFlow, 310
RateForBalance, 310
RateForFlow, 311
ReviewStatus, 311
ReviewStatusUsingPhaseID, 312
Round, 313
Scale, 314
SecurityAsPartner, 315
SecurityClass, 316
SetData, 317
SetDataWithPOV, 318
SubmissionGroup, 319
SupportsTran, 319
SwitchSign, 319
SwitchType, 321
Trans, 322
TransPeriodic, 322
UD1...3, 323
ValidationAccount, 324
ValidationAccountEx, 325
XBRLTags, 325
HDE files, defining, 195
Holding function, 280
HoldingCompany attribute, 82
HS object, 222

ICP, 94
ICP members
  editing, 95
  system generated, 94
ICPEntitiesAggregationWeight attribute, 86
ICPTopMember attribute, 76
ICPTopMember function, 280
ICPWeight function, 281
IDFromMember function, 282
ImpactStatus function, 283
Input function, 284
Instructions keyword, Web scripts, 159
Intercompany dimension, 18
intercompany matching reports
  selecting member lists, 198
  selecting style sheets, 199
  specifying currencies, 199
  specifying decimal places, 199
  suppression options, 199
Intercompany Partner Matching reports, 198
intercompany partners, setting up, 94
intercompany transactions
  adding reason codes, 331
  alerting, 62
  closing periods, 329
  editing reason codes, 331
  locking entities, 329
  opening periods for, 327
  removing reason codes, 332
  unlocking entities, 329
invalid characters in labels and descriptions, 18
IsAlmostEqual function, 284
IsBase function, 285
IsCalculated attribute
  accounts, 76
  custom, 80
IsCalculated function, 286
IsChild function, 287
IsConsolidated attribute, 76
IsConsolidated function, 289
IsDescendant function, 289
IsFirst function, 291
IsHoldingMethod attribute, 89
IsICP attribute
  accounts, 76
  entities, 82
IsICP function, 292
IsLast function, 293
IsTransCur function, 293
IsTransCurAdj function, 294
IsValidDest function, 294
IsZero function, 295

JLF files
  creating, 137
  extracting in Web, 144
  extracting in Windows, 143
  loading in Web, 143
  loading in Windows, 142
journal files (JLF)
  creating, 137
  extracting in Web, 144
  extracting in Windows, 143
  label requirements for, 18
  loading in Web, 143
  loading in Windows, 142
journal reports
  creating, 197
  example, 197

label requirements, 18
LAST keyword, Web scripts, 175
launching user management, 56
liability account behavior, 78
limitations for characters, 18
LineItemDetailSinglePeriod keyword, Web scripts, 159
Link keyword, Web scripts, 159
List function, 296
List View
  adding and modifying members, 113
  copying, cutting, and pasting members, 114
  deleting members, 114
  showing and hiding columns, 115
sorting metadata, 115
validating metadata, 115
loading
data form scripts (WDF), 183
in Web
application security files (SEC), 64
journal files (JLF), 143
member list files (LST), 133
metadata files (XML and APP), 121
rules files (RLE), 231
in Windows
application security files (SEC), 63
journal files (JLF), 142
member list files (LST), 132
metadata files (XML and APP), 120
rules files (RLE), 230
rules in Expression Builder, 228
logging off users on system, 42
LST files
creating, 127
extracting in Web, 134
extracting in Windows, 133
loading in Web, 133
loading in Windows, 132
currencies example, 99
custom dimension example, 101
dimensions in metadata file, 99
dimensions not in the metadata file, 105
etentities example, 101
file format, 107
members in metadata file, 100
scenarios example, 100
metadata file
account dimension hierarchies section, 104
consolidation methods section, 103
custom dimension hierarchies section, 105
entity dimension hierarchies section, 104
hierarchies section, 103
scenario dimension hierarchies section, 104
metadata files
extracting in Web, 124
extracting in Windows, 123
loading in Web, 121
loading in Windows, 120
metadata files (APP or XML), creating, 97
metadata integrity, 118
Metadata Item Properties
creating aliases for Custom dimensions, 116
Metadata Manager
File Properties, creating metadata reports, 116
Metadata Item Properties, creating aliases for Custom dimensions, 116
views defined, 106
Metadata Manager List View
adding and modifying members, 113
copying, cutting, and pasting members, 114
deleting members, 114
showing and hiding columns, 115
sorting metadata, 115
validating metadata, 115
Metadata Manager Tree View
adding and modifying members, 108
adding members, 109
adding orphaned members, 111
deleting and removing members, 111
expanding and collapsing the hierarchy, 112
modifying node attributes, 109
moving members, 110
promoting members, 110
removing orphaned members, 112
repositioning members, 110

M
MaxCells keyword, Web scripts, 160
MaxColsForSparseRetrievalMethod keyword, Web
scripts, 160
MaximumReviewLevel attribute, 84
Member attribute
accounts, 77
custom, 81
entities, 82
scenarios, 84
Member function, 297
member list files (LST)
creating, 127
extracting in Web, 134
extracting in Windows, 133
loading in Web, 133
loading in Windows, 132
member list files (LST), dynamic lists, 130
MemberFromID function, 298
metadata
accounts example, 100
application settings example, 99
method account, 93
Method function, 299
moving periods, 30

N
NEXT keyword
   Web scripts, 175
NodeSecurity attribute, 87
NoInput function, 299
NoRound function, 300
NoSuppress keyword, Web scripts, 160
NumBase function, 301
Number function, 302
NumChild function, 303
NumDecimalPlaces attribute, 77
NumDecimals keyword
   Web scripts, 161
NumDescendant function, 304

O
objects, HS, 222
OpenDataUnit function, 306
opening applications, 35
organization by period, 88
OrgByPeriodApplication attribute, 87
Override keyword
   Web scripts, 161
ownership accounts
   SharesOutstanding account, 93
   SharesOwned account, 93
   VotingOutstanding account, 93
   VotingOwned account, 93

P
PCON account, 93
PCon function, 307
PCTRL account, 93
PER files, creating, 25
percent consolidation, 93
percent control, 93
percent direct ownership, 93
percent ownership, 93
PercentConsol attribute, 89
Period dimension, 17
Periodic frequency, 27
periods
copying, 31
deleting, 31
moving, 30
PlugAcct attribute, 77
PlugAcct function, 307
POWN account, 93
POwn function, 308
PrintNumColsPerPage keyword, Web scripts, 162
PrintNumRowsPerPage keyword, Web scripts, 163
PrintRepeatHeadersonAllPages keyword, Web scripts, 163
process management
   alerting, 61
   submission phases, 351
PVAForBalance function, 309
PVAForFlow function, 310

R
RateForBalance function, 310
RateForFlow function, 311
ReadOnly keyword, Web scripts, 165
reason codes for intercompany transactions, 330
records, deleting invalid, 51
referential integrity, 118
report definitions
   Data Explorer report, 196
   Intercompany Partner Matching Reports, 198
   Journal report, 197
ReportDescription keyword, Web scripts, 165
ReportLabel keyword, Web scripts, 165
reports
   Data Explorer reports, 196
   Intercompany Partner Matching reports, 198
   Journal reports, 197
ReportSecurityClass keyword
   Web scripts, 166
ReportType keyword, Web scripts, 167
requirements for user-defined elements, 18
revenue account behavior, 78
ReviewStatus function, 311
ReviewStatusUsingPhaseID function, 312
RLE files
   creating, 229
   extracting in Web, 232
   extracting in Windows, 231
   loading in Web, 231
   loading in Windows, 230
Rn keyword, Web scripts, 164
Round function, 313
RPT files, defining, 195
rules
  account expressions, 218
  loading, 228
  types, 212
Rules Editor
  configuring, 227
  creating rules, 227
rules files (RLE)
  creating, 229
  extracting in Web, 232
  extracting in Windows, 231
  loading in Web, 231
  loading in Windows, 230
running security reports, 62

S
saving application profiles, 32
SCalc keyword
  Web scripts, 167
Scale attribute, 91
Scale function, 314
Scale keyword
  Web scripts, 168
scenario attributes, 83
Scenario dimension, 17
Scenario View frequency, 27
scenarios example, 100
SEC files
  extracting in Web, 69
  extracting in Windows, 68
  loading in Web, 64
  loading in Windows, 63
security classes
  assigning access, 59
  security reports, 62
SecurityAsPartner attribute, 83
SecurityAsPartner function, 315
SecurityClass attribute
  accounts, 77
  custom, 81
  entities, 83
  scenarios, 84
SecurityClass function, 316
SelectablePOVList keyword, Web scripts, 168
SetData function, 317
SetDataWithPOV function, 318
setting up, 94
setting up alerting
  intercompany transactions, 62
  process management, 61
setting up ICP, 94
Shared Services applications
  overview, 336
Shared Services models
  access permissions, 345
  application system members, 343
  assigning permissions, 347
  compare operations, 341
  comparing, 340
  configuring for external authentication, 335
  deleting, 344
  deleting permissions, 347
  described, 335
  dimensional hierarchies, 335
  export operations, 339
  exporting, 338
  Manage Models user role, 336
  managing permissions, 345
  naming restrictions, 340
  non-dimensional hierarchies, 335
  permissions, 345
  properties, viewing and setting, 348
  registering applications, 335
  renaming, 344
  setting properties, 348
  sharing data, 336
  sharing metadata, 336
  system members, 343
  tracking version history, 344
  types of permission, 345
  user authentication, 345
  user roles, 336
  versioning, 344
  viewing, 337
  viewing content, 342
  viewing properties, 348
SharesOutstanding account, 93
SharesOwned account, 93
ShowDescription keyword, Web scripts, 169
ShowLabels keyword, Web scripts, 170
specifying
frequencies in applications, 27
languages in applications, 25
periods in applications, 28
star schema, 188
creating and exporting data to, 191
deleting, 193
updating, 192
star schema template
creating, 193
deleting, 194
String keyword, Web scripts, 170
Style keyword, Web scripts, 170
Submission Group function, 319
SupportsProcessManagement attribute, 84
SupportsTran function, 319
SuppressColHeaderRepeats keyword, Web scripts, 172
SuppressInvalidColumns keyword
Web scripts, 173
SuppressInvalidRows keyword
Web scripts, 173
SuppressNoDataColumns keyword, Web scripts, 174
SuppressNoDataRows keyword, Web scripts, 174
SuppressZeroColumns keyword, Web scripts, 174
SuppressZeroRows keyword, Web scripts, 175
SwitchSign function, 319
SwitchSignForFlow attribute, 81
SwitchType function, 321
SwitchTypeForFlow attribute, 81
syntax
data form scripts, 147
system accounts
active account, 93
Consol1...3 account, 93
DOWN account, 93
editing, 94
method account, 93
PCON account, 93
PCTRL account, 93
POWN account, 93
SharesOutstanding account, 93
SharesOwned account, 93
VotingOutstanding account, 93
VotingOwned account, 93
T
task audit, 45
ToPercentControl attribute, 90
ToPercentControlComp attribute, 90
Trans function, 322
translation default, 221
translation, default, 220
TranslationOperator attribute, 92
TransPeriodic function, 322
Tree View
adding and modifying members, 108
adding members, 109
adding orphaned members, 111
deleting and removing members, 111
expanding and collapsing the hierarchy, 112
modifying node attributes, 109
moving members, 110
promoting members, 110
removing orphaned members, 112
repositioning members, 110
U
UD1...3 function, 323
updating a star schema, 192
UsedByCalcRoutine attribute, 90
UsePVAForBalanceAccounts attribute, 87
UsePVAForFlowAccounts attribute, 87
user and group assignments, 57
user management console, 56
user-defined element requirements, 18
UserDefined1...3 attribute
custom, 82
entities, 83
scenarios, 85
users on system
disabling connections, 44
enabling connections, 44
logging off users, 42
viewing users, 42
UseSecurityForAccounts attribute, 87
UseSecurityForCustom attribute, 88
UseSecurityForEntities attribute, 88
UseSecurityForICP attribute, 88
UseSecurityForScenarios attribute, 88
UsesLineItems attribute
scenarios, 85
V
ValidationAccount attribute, 88
ValidationAccount function, 324
ValidationAccountEx function, 325
Value dimension, 18
value members
   editing, 96
   system generated, 94
View dimension, 18
viewing users on system, 42
views, corresponding frequencies and, 27
VotingOutstanding account, 93
VotingOwned account, 93

W
WDF files
   creating, 147
   extracting, 184
   loading, 183
Web form builder
   Columns tab, 179
   Details tab, 180
   POV tab, 178
   Properties tab, 178
   Rows tab, 180
Web scripts
   AddMember keyword, 150
   BackgroundPOV keyword, 151
   Blank keyword, 152
   Calc1 keyword, 153
   CalcByRow keyword, 153
   Cell_Link keyword, 154
   CellText keyword, 154
   Cn keyword, 152
   CUR, FIRST, LAST, NEXT keywords, 175
   CustomHeader keyword, 155
   CustomHeaderStyle keyword, 156
   DynamicPOV keyword, 156
   FormInputBoxLength keyword, 156
   FormNumDecimals keyword, 157
   FormRowHeight keyword, 157
   FormScale keyword, 157
   Instructions keyword, 159
   LineItemDetailSinglePeriod keyword, 159
   Link keyword, 159
   MaxCells keyword, 160
   MaxColsForSparseRetrievalMethod keyword, 160
   NoSuppress keyword, 160
   NumDecimals keyword, 161
   Override keyword, 161
   PrintNumColsPerPage keyword, 162
   PrintNumRowsPerPage keyword, 163
   PrintRepeatHeaderOnAllPages keyword, 163
   ReadOnly keyword, 165
   ReportDescription keyword, 165
   ReportLabel keyword, 165
   ReportSecurityClass keyword, 166
   ReportType keyword, 167
   Rn keyword, 164
   SCalc keyword, 167
   Scale keyword, 168
   SelectablePOVList keyword, 168
   ShowDescription keyword, 169
   ShowLabels keyword, 170
   String keyword, 170
   Style keyword, 170
   SuppressColHeaderRepeats, 172
   SuppressInvalidColumns keyword, 173
   SuppressInvalidRows keyword, 173
   SuppressNoDataColumns keyword, 174
   SuppressNoDataRows keyword, 174
   SuppressRowHeaderRepeats keyword, 174
   SuppressZeroColumns keyword, 174
   SuppressZeroRows keyword, 175
   Workspace, 16

X
XBRLTags attribute, 78
XBRLTags function, 325
XML to APP, 107

Y
Year dimension, 17

Z
ZeroViewForAdj attribute, 85
ZeroViewForNonadj attribute, 85