Oracle® Fusion Cloud EPM

Designing with Calculation Manager

E96249-61

Oracle Fusion Cloud EPM Designing with Calculation Manager,

E96249-61

Primary Author: EPM Information Development Team

Contents

9 Using Member Selection, Variables, Functions, Smart Lists, and Planning Formula Expressions to Design Components

Documentation Accessibility

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

Access to Oracle Support

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

Documentation Feedback

To provide feedback on this documentation, click the feedback button at the bottom of the page in any Oracle Help Center topic. You can also send email to epmdoc_ww@oracle.com.

1

Creating and Running an EPM Center of Excellence

A best practice for EPM is to create a CoE (Center of Excellence).

An **EPM CoE** is a unified effort to ensure adoption and best practices. It drives transformation in business processes related to performance management and the use of technology-enabled solutions.

Cloud adoption can empower your organization to improve business agility and promote innovative solutions. An EPM CoE oversees your cloud initiative, and it can help protect and maintain your investment and promote effective use.

The EPM CoE team:

- Ensures cloud adoption, helping your organization get the most out of your Oracle Fusion Cloud EPM investment
- Serves as a steering committee for best practices
- Leads EPM-related change management initiatives and drives transformation

All customers can benefit from an EPM CoE, including customers who have already implemented EPM.

How Do I Get Started?

Click to get best practices, guidance, and strategies for your own EPM CoE: Introduction to EPM Center of Excellence.

Learn More

- Watch the Cloud Customer Connect webinar: Creating and Running a Center of Excellence (CoE) for Cloud EPM
- Watch the videos: Overview: EPM Center of Excellence and Creating a Center of Excellence.
- See the business benefits and value proposition of an EPM CoE in *Creating and Running an EPM Center of Excellence*.

2

Overview of Calculation Manager

Related Topics

- About Calculation Manager
  Use Calculation Manager to create, validate, deploy, and launch calculations that solve business problems.
- Services That Use Calculation Manager
  Several Enterprise Performance Management Cloud services use Calculation Manager.
- Launching Calculation Manager
  Launch Calculation Manager to create rules, rulesets, components, and templates.
- Viewing Objects in Calculation Manager
  Views allow you to see Calculation Manager objects in different contexts.
- Customizing the Columns in a View
  You can customize the columns that are displayed in each view, and the order in which the columns are displayed.
- Filtering Objects
  Filter objects in Calculation Manager in System View or Filter View.
- Using Queries to Filter Objects
  In Filter View, after you filter objects, you can use queries to further refine the objects that are displayed.
- Looking at a Graphical Picture of a Rule, Component, or Template
  Rules, components, and templates are displayed graphically in a flow chart within the Rule Designer and the Template Designer.

About Calculation Manager

Use Calculation Manager to create, validate, deploy, and launch calculations that solve business problems.

You can create the following types of objects in your calculations:

- **Rules**—Objects containing components, templates, and other rules
- **Rulesets**—Objects containing two or more business rules that can be calculated simultaneously or sequentially
- **Components**—Objects containing formulas, scripts, conditions, member and data ranges, fixed loops, and design-time prompts. (Components are not deployable.)
- **Templates**—Objects that you can use in business rules to perform a calculation or a set of calculations

Services That Use Calculation Manager

Several Enterprise Performance Management Cloud services use Calculation Manager.

- Planning

- Planning Modules
- Enterprise Profitability and Cost Management
- Financial Consolidation and Close

Launching Calculation Manager

Launch Calculation Manager to create rules, rulesets, components, and templates.

To launch Calculation Manager:

1. In the top left corner of the Planning Home page, click  .
2. Under **Create and Manage**, click **Rules**.

Viewing Objects in Calculation Manager

Views allow you to see Calculation Manager objects in different contexts.

When you are in a view, you can use the options on the **View** menu to customize the columns that display and the order in which they display.

When you launch Calculation Manager, the **System View** is automatically displayed. To switch views, select a view from the drop-down next to **Select View**.

Calculation Manager includes the following views:

- **System View**—Default view displayed when you launch Calculation Manager. It lists all the applications and objects to which you have access.

  Your access privileges are determined by the role you are assigned in Application Management. Access privileges are assigned on an application basis.
- **Custom View**—Create folders and add objects to the folders to create a view that is meaningful to you.

  To create a folder in **Custom View**:

  1. In the right pane, right-click an application, then select **New**, and then **Folder**.
  2. In **New Folder**, enter a folder name, and then click **OK**.
  3. Add objects to the folder by dragging the objects from the **Existing Objects** pane and dropping them into the folder.
- **Deployment View**—Lists, by application type and application, the rules and rule sets that are deployable and their deployment and validation status. You can select what rules and rule sets to make deployable, and then deploy the objects to applications.

  Deploying one or more rules and rules sets in an application is known as a partial deployment, Deploying all rules and rule sets in an application is known as a full deployment.
- **Filter View**—Filter the objects that are displayed.

  Use to You can define filter options in the **Filter** dialog box, and then use a query to further refine the filter.

Customizing the Columns in a View

You can customize the columns that are displayed in each view, and the order in which the columns are displayed.

To customize the columns in a view, do one of the following:

- Select **View**, then **Columns**, and then select the columns to display. Reorder the columns by selecting **View**, then **Reorder Columns**, and then select the column order.
- Select **View**, then **Columns**, and then **Manage Columns**. In the **Mange Columns** dialog box, select which columns to display and the order in which to display them.

> **Note**
>
> In Oracle Financials Cloud, there is a new "Unlock" feature for columns. This column is not shown by default. Unlock allows an administrator to unlock an object that is locked by another administrator.

Filtering Objects

Filter objects in Calculation Manager in System View or Filter View.

You can filter objects by application type, application, calculation type, plan type, database, object type (business rules, business rule sets, formula and script components, and templates), and deployment or validation status.

To filter objects:

1. In **System View** or **Filter View**, click  .
2. In the **Filter** dialog box, enter the requested information, and then click **OK**.

Using Queries to Filter Objects

In Filter View, after you filter objects, you can use queries to further refine the objects that are displayed.

To use queries to filter objects:

1. In **Filter View**, click  , and then enter information in the **Filter** dialog box.
2. Click  to display text boxes above the columns.
3. In the text boxes above the columns, enter a query to further refine the filter.

   You cannot use wild cards or partial text strings. To search for a plan type, enter the starting characters of the first word. For example, if a plan type is named "Plan1", and you enter "la" in the query, only objects that begin with "La" are displayed. In this example, to get the desired result, you would enter "Pl" to display all plan types that start with "Pl".

> **Note**
>
> You can also use queries to filter objects in the Variable Designer.

Looking at a Graphical Picture of a Rule, Component, or Template

Rules, components, and templates are displayed graphically in a flow chart within the Rule Designer and the Template Designer.

When you open a rule or template, you can select its components (for example, formulas, scripts, conditions, ranges, and loops) in the flow chart to see details. You can also increase or decrease the size of the flow chart to view or hide component details.

When you select a component in a flow chart, its properties, uses, and other information are displayed in tabs below the flow chart. As you move among the components, the tabs below the flow chart change.

For example, if you open a business rule that contains a formula component and a script component, and select the formula component in the flow chart, the properties of the formula (name, description, application and application type) are displayed in the tabs below the flow chart. If you then select the script component in the flow chart, the text, properties, and usages of the script component are displayed in the tabs below the flow chart.

3

Designing Business Rules

Related Topics

- About Business Rules
  Calculation Manager enables you to create, validate, deploy, and administer sophisticated multidimensional business rules.
- Best Practices for Designing Business Rules
- Creating a Business Rule
  A business rule is a Calculation Manager object that consists of calculations grouped into components.
- Creating a Groovy Business Rule
  Oracle supports the creation of business rules written in the Groovy scripting language.
- Opening a Business Rule
  You open a business rule from the System View that is displayed by default when you open Calculation Manager.
- Editing Business Rules
- Saving Business Rules
  Save business rules to the application and application type for which they were created.
- Running a Business Rule
  You must open a business rule before you run it.
- Copying Business Rules
  You can copy a business rule to another application and plan type.
- Searching in Business Rules
  Search for a text string in a business rule script. Search for and replace text in a graphical business rule.
- Printing a Business Rule
  You can print a business rule's properties, its flow chart, and the details of its components.
- Deleting a Business Rule
  You can delete a business rule only if it is not used by other rules or rulesets.
- Defining Common Dimensions In Business Rule Components
  Define common dimensions by opening the business rule and selecting the members, variables, and functions that are common for each dimension.
- Analyzing and Debugging Business Rules
  Analyze a business rule by running the rule and capturing statistical information. Debug a business rule by running the rule and examining its script.
- Refreshing Business Rules or Business Rulesets
  In the System View, Custom View, and Deployment View, you can refresh any level of the application list.
- Refreshing Metadata Changes
- Showing the Usages of a Business Rule or Ruleset
  Display the rules, templates, and rulesets that are using a business rule or business ruleset.

- Optimizing Business Rules
  Leverage available tools and effectively manage your rules.

About Business Rules

Calculation Manager enables you to create, validate, deploy, and administer sophisticated multidimensional business rules.

You typically create business rules to:

- Allocate costs among entities
- Model revenues
- Model expenses
- Prepare a balance sheet
- Calculate cash flow
- Calculate currency translation adjustments
- Calculate group and minority interest
- Calculate deferred taxes

Before you create a business rule, you should be familiar with the database outline and the application with which you are working. Having this information helps you to create business rules more efficiently. You should also understand the following about your data:

- How the data is stored and aggregated
- At what level the data gets loaded into the database
- Calculation order
- Key assumptions that drive the calculations

You can create business rules using components such as formulas, scripts, loops, data and member ranges, templates, and variables, including runtime prompt variables. (See Using Components to Design Business Rules and Templates.)

As you create business rules, you can leave the components, templates, and variables you are working with open. Calculation Manager displays these objects in a tabbed interface so you can move easily among the tabs as you create the rules. You can have as many as ten tabs open. For optimum performance, however, you should not open more than ten objects simultaneously.

To launch a business rule in Planning, the administrator must give launch privileges to the rule.

> **Note**
>
> You can also create a business ruleset of two or more related rules (or rulesets) that you can launch simultaneously or sequentially. See Designing Business Rule Sets.

Best Practices for Designing Business Rules

Poorly written rules have a major impact on all aspects of an application. Below are some best practices when designing your business rules. Following best practices recommendations can lead to significant performance benefits.

Follow these key design guidelines for business rules.

Top 10 Best Practices for Rules

Follow these best practice recommendations when designing your business rules. These guidelines can lead to significant performance benefits because poorly written rules have a major impact on all aspects of an application.

1. Follow these guidelines for SET commands:
   - At the top of the rule, do not use `SET CREATEBLOCKONEQ ON` or `SET CREATENONMISSINGBLK ON`.
   - Do not use administrative type commands like this in end-user rules, as this requires a restructure: `SET CLEARBLOCK EMPTY`.
   - Avoid or test rules using `SET CALCTASKDIMS`. (Oracle Essbase typically does this automatically.)
   - The Data Copy rule should include the following to prevent copying empty blocks within the Fix statement: `SET COPYMISSINGBLOCK OFF`
2. Block Creation should be done using either Datacopy or Sparse Member assignment. The functions `@createblockoneq` and `@createblock` should be used as a last resort inside a limited Fix statement.
3. Avoid missing dimension references in a Fix statement (for example, in Data Copy Rule). This can waste processing time and increase contention and create unnecessary blocks for all levels of the missing dimensions.
4. Remove parallel calculation in business rules associated with forms. Calc Parallel or Fix Parallel should be used only with administrative/batch rules.
5. Do not create unnecessary zeros, as this leads to block and data explosion. Carefully review business logic and add necessary If conditions to check for zeros. Convert zeros to `#missing`. After this, a dense restructure is required to remove the blocks.
6. Eliminate multiple passes on the same blocks; instead, set a proper outer fix and move in and out as necessary. Combine If statements instead of using and re-using If on the same intersections.
7. Avoid using cross-dimensional references on the left side of an equation. This has a performance impact.
8. Aggregate dimensions in order of creation of the most blocks to the least blocks in the script, for example, Agg (1st Most Blocks Dimension, 2nd Most Blocks Dimension, 3rd Most Blocks Dimension). Agg is faster than Calc Dim and is a preferred mechanism for aggregation. Aggregation using `@ancestors` in end user rules all the way to the top of the dimension can lead to block contention.
9. Use run-time prompts instead of creating multiple rules with the same underlying logic. More rules means more maintenance.

10. Use templates for breaking down and reusing business logic. However, templates should not be fully functioning rules with Fix, EndFix. The rule combining various templates should have a proper outer Fix and move in and out of smaller pieces as necessary.

Add Business Logic Using Calculations

To incorporate your business logic into your application, you can build calculations using Calculation Manager. This lets you create, validate, deploy, and administer sophisticated calculations that solve business problems.

You typically create business rules and rulesets to:

- Perform revenue modeling
- Perform expense modeling
- Calculate KPIs
- Perform allocations

Calculation Manager includes these objects:

- Rules: Contain components and templates
- Components: Assist you in building rules
- Rulesets: Contain rules that can be calculated simultaneously or sequentially
- Templates: Include system templates that perform calculations and custom templates that can be designed by administrators

To learn more about creating calculations, see the guidelines in Designing with Calculation Manager for Oracle Enterprise Performance Management Cloud.

Build Aggregations

Aggregations roll up your application to summary-level members in the dimension, such as Entity or any other sparse dimension.

Calculation Manager includes templates to help you build aggregations. Here are some suggestions on how to use templates.

Set the Point of View

When the point of view is set, the rule will run only for the selected members. Using a runtime prompt for the dimensions allows users to specify member values for these dimensions when launching the rule. This way, users can launch the rule several times for different years, scenarios, and versions without having to modify the rule in Calculation Manager.

Typical settings:

- Full dense aggregation: Complete this section if parent values in your dense dimensions are not set to dynamic calc. Typically this tab is left empty.
- Full sparse aggregation: Select the sparse dimension that needs to be aggregated. The order of the selected dimensions is not relevant.
- Partial dimension aggregation, Dense: Complete this section if parent values in your dense dimensions are not set to dynamic calc. Typically this tab is left empty.
- Aggregate the data up to the local currency: No
- Aggregate the missing values in the Database: Yes

  Be careful when using this option with a Target Version where data is entered at the parent members and descendants will be #Missing.

- Optimize the Calculation on Sparse dimension: Off
- Select a value for the calculator cache: Default
- Do you want to activate the debug mode for this wizard?: Debug Wizard On or Debug Wizard Off. Select Debug Wizard On if you want to see a script generated to display selections for some of the Design Time Prompts in this template.

Best practices:

- Leverage runtime prompts for members such as Entity, Scenario, and Version. This allows your rule to be dynamic and run based on user input.
- Typically, dense dimensions such as Account and Period don't need to be aggregated. If this is the case, you can set parent members to dynamic calc. However, if you have member formulas on dense dimensions and they are not set to dynamic calc, a Calc Dim rule is required.

Build Detailed Calculations

You use Calculation Manager to create, validate, deploy, and administer calculations that solve business problems.

There are three types of objects that can be calculated in Calculation Manager:

- Rulesets: Contain rules that can be calculated simultaneously or sequentially
- Rules: Contain components and templates
- Components: Contain formula components, script components, condition components, range components, and fixed loop components

Best practices:

- As a first step in building your rules, ensure that you understand the business logic and which entities or departments the rule applies to. For example, know the accounts that are involved in the rule.
- Be sure you know the source and destination accounts.
- After you fully understand the drivers of the calculation, use the proper object component or template to build the rule. The components and templates facilitate member selection to help deploy the rules.
- Leveraging runtime prompts for members such as Entity, Scenario, and Version allows your rules to be dynamic and run based on user input.

Calculation Manager Diagnostics

Run errors and warnings before deploying rules. This provides helpful information, including:

- The number of passes through the database
- Any necessary warnings
- Information on the number of blocks and if dimensions are missing
- Any rules that need to be optimized
- If any of the components of the right side of the equation contain a zero, the derived member will be 0. Then after aggregation there are many 0's.

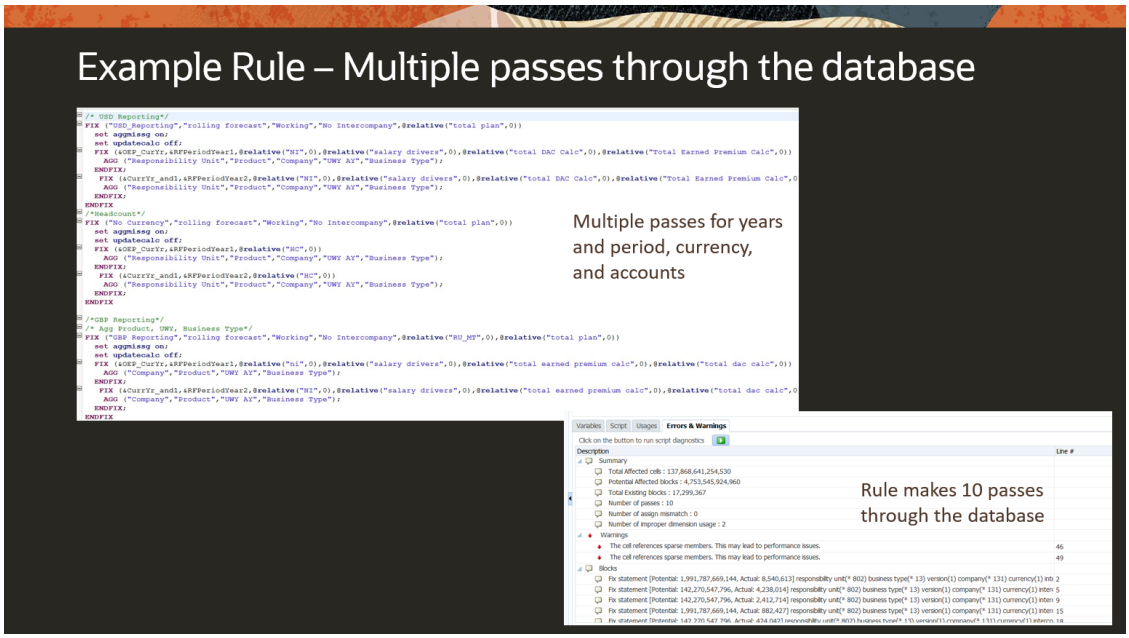
- To address this, the rule should include an if statement such as the following. This way, zeros will not be so significant in the application.

```
if ("Earned Premium"<>0)
```

Example Rule Issues and Solutions

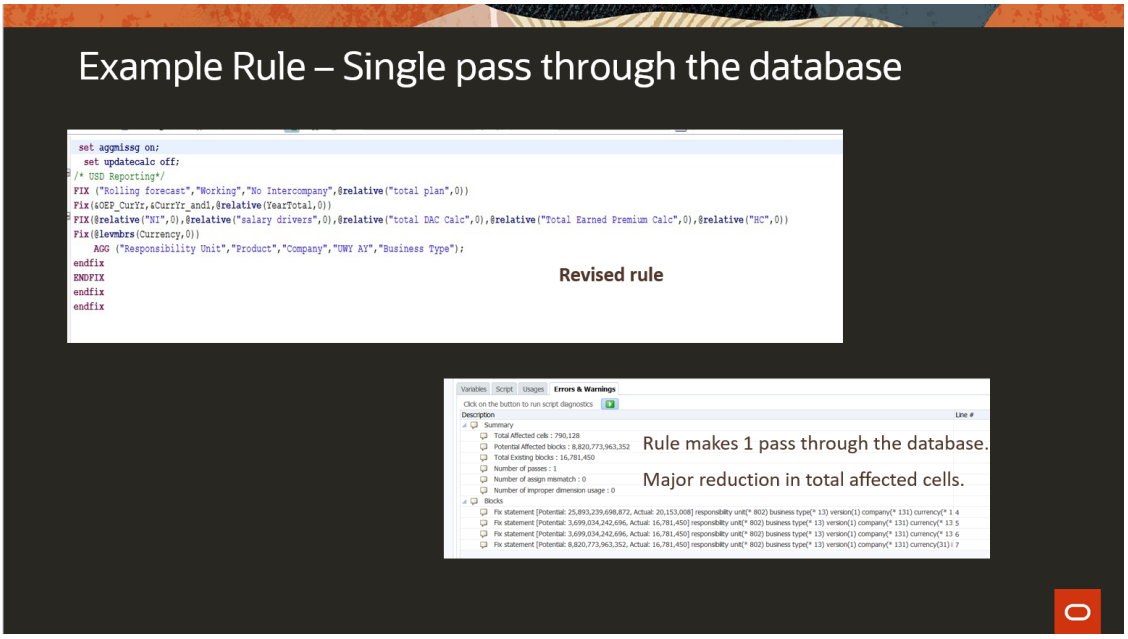
Example 1: Rule that Makes Multiple Passes Through the Database

The following rule makes 10 passes through the database for years and period, currency, and accounts.



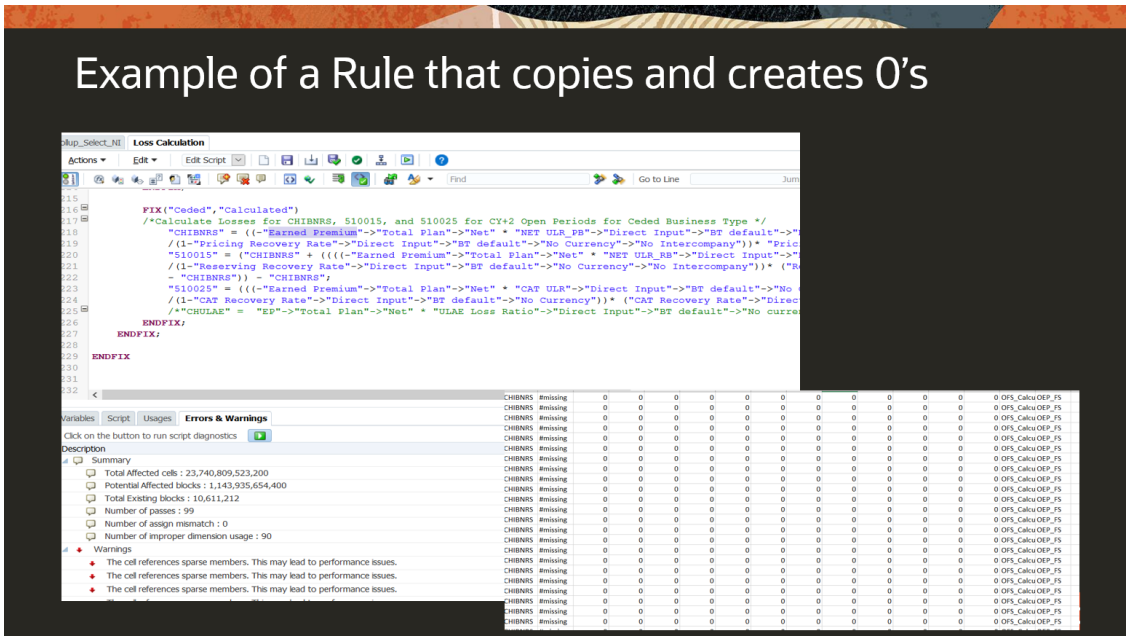
Example 2: Rule that Makes Only One Pass through the Database

The following rule makes just one pass through the database. This is a major reduction in the total affected cells.



Example 3: Rule that Copies and Creates Zeros

In this rule, if any of the components of the right side of the equation contain a 0, the derived member will be 0. After aggregation, there will be many zeros. To address this, the rule should include an if statement saying `if ("Earned Premium"<>0)`. This way zeros would will not be so significant in the application.



Creating a Business Rule

A business rule is a Calculation Manager object that consists of calculations grouped into components.

A rule can contain one or more components, templates, or rules.

You can create business rules for applications to which you have access. Your ability to create rules is determined by the role you are assigned. (See *Administering User Provisioning for Oracle Enterprise Performance Management Cloud*).

Rules are represented graphically in a flow chart into which you can drag and drop components to design the rule.

To create a business rule:

1. Do one of the following:

   - In **System View**, **Custom View**, **Deployment View**, or **Filter View,** click  , and then enter the information in the **New Object** dialog box. Make sure to select **Rule** as the **Object Type**.
   - In **System View**, right-click **Rules**, then select **New**, and then enter the information in the **New Rule** dialog box.

   > **Note**
   >
   > The following characters are not allowed in the rule name: '\r', '\n', '\t', '\f', '\b', '<', '>', '(', ')', '"', '\\', '{', '}', '[', ']', '*', '?'

2. In the Rule Designer, select objects under **New Objects** and **Existing Objects**, and then drop them into the flow chart between **Begin** and **End**.

   Note the following:

   - When you drag an existing formula or script component into the flow chart, by default, the formula or script becomes a shared object. If you do not want it to be shared, clear the **Shared** check box in the formula or script component's **Properties**. See Sharing Script and Formula Components.
   - To view a script component in its graphical format, right-click the script component in the flow chart, and then select **Convert to Graphical**.

     A script component is converted to its graphical format only if the script is valid. To undo the conversion to graphical format, right-click the script component, and then select **Undo**.
   - You can create objects like formulas and scripts independently of the rule, and add them to the rule later.
   - To work with a business rule in its non-graphical format (its script format), click the drop-down next to **Designer**, and then select **Edit Script** (See Editing a Business Rule in Script Mode.)

3. Enter **Properties** for the rule.

   The properties change as you add components to the rule and move among the components in the flow chart. To enter properties for a specific component, select the component in the flow chart.

   The following properties are displayed when you select **Begin** or **End** in the flow chart.

   - **General**: Name, description, and comments
   - **Location**: Application and plan type
   - **Options**: Options specific to the current application

Table 3-1 Options

Property	Description
Create Dynamic Members	Create members when you specify a default dynamic parent in a variable of the member type with runtime prompts enabled. If you create dynamic members in a business rule, and select a default dynamic parent, the children members of the parent are automatically created *before* the rule is launched in your application. **Note:** If you select Create Dynamic Member, the newly-created members are deleted if the business rule fails to launch. **Note:** This property is not available for Financial Consolidation and Close applications.
Delete Dynamic Members	Delete members when you specify a default dynamic parent in a variable of the member type with runtime prompts enabled. If you delete dynamic members in a business rule, and select a default dynamic parent, the children members of the parent are automatically deleted *after* the rule is launched in your application. **Note:** This property is not available for **Financial Consolidation and Close** applications.
Enable Notifications	Enable a business rule to send an email notification to a logged on user when the rule launches with or without errors.

4. Enter or review information on the following tabs in the Rule Designer:
   - **Global Range**: Define common dimensions in business rule components.

     See Defining Common Dimensions In Business Rule Components.
   - **Variables**: Review and define information about the variables used in the business rule.

     > **Note**
     > - The **Variables** tab displays only when the business rule contains runtime prompt values.
     > - For non-groovy rules, variables of type *member* or *members* are the only supported types of variables for Planning cubes of type *Aggregate Storage Option* (ASO)
     > - When you launch a rule from Calculation Manager, following options are *not* honored:
     >   * **Is Hidden**
     >   * **Security**
     >   * Any valid intersections defined in Planning

   - **Script**: View the generated script for the rule.

     You cannot make changes on this tab. To make changes to the script, select **Edit Script** in the drop down next to **Designer**.
   - **Usages**: View which rules and rulesets use the rule.

     You cannot edit the information on this tab. By default, a rule is not used by any rules or rulesets when you create it.

- **Errors and Warnings**: Click  to run script diagnostics for the business rule. When you do this, Calculation Manager analyzes the business rule script, and displays either:
  - Validation errors, if the rule does not validate.

    To fix a validation error, right-click the error, and then select **Show in Script** or **Show in Designer**. When you select **Show in Designer**, Calculation Manager displays the component with the error, where you can make the necessary changes, save the rule, and rerun script diagnostics. To edit the component, you must be in the Designer view.
  - Summary, Warnings, and Blocks, if the rule validates.

    Click on a warning or block item, and then select **Show in Script** or **Show in Designer**. You can only edit the components in the Designer view.
    * **Summary**: Statistics such as the number of data cells in the calculation, the number of passes through the data, and the number of dimensions that are used incorrectly.
    * **Warnings**: Information such as whether all sparse dimensions are specified for cell references, whether a cell references sparse members, and whether an assignment references sparse dimension members in different data blocks.
    * **Blocks**: Information such as: "For each Fix statement, what is the potential and estimated number of blocks that could be affected by the fix statement?"

5. Click  to save the rule.

Creating a Groovy Business Rule

Oracle supports the creation of business rules written in the Groovy scripting language.

Related Topics

- About Groovy Business Rules
  Groovy business rules allow you to design sophisticated rules that solve use cases that normal business rules can't solve; for example, rules to prevent users from saving data on forms if the data value is above a predefined threshold.
- Creating a Groovy Business Rule for ASO Cubes
  You can create a Groovy business rule for a ASO cube.
- Creating a Groovy Business Rule for BSO Cubes
  You can create a Groovy business rule for a BSO cube.
- Editing the Script For a Groovy Business Rule or Template
  You can edit the script for a Groovy rule or template.
- Validating Groovy Scripts
  Validate Groovy business rule scripts to ensure that your application's Groovy scripts can pass the stricter validation rules.
- Resolving Groovy Business Rule Validation Issues
- Java API Reference for Groovy Rules
  For Enterprise applications, a Java API Reference is available to use as you create Groovy rules.
- Groovy Business Rule Examples
  Example Groovy scripts are available.

- Groovy Business Rule Tutorial Videos
  Watch these tutorial videos for details and best practices when implementing and using Groovy Business Rules.
- Groovy Business Rule Tutorials
  Complete these tutorials for hands-on examples on implementing Groovy Business Rules.

About Groovy Business Rules

Groovy business rules allow you to design sophisticated rules that solve use cases that normal business rules can't solve; for example, rules to prevent users from saving data on forms if the data value is above a predefined threshold.

> **Note**
>
> Groovy is an advanced customizable rules framework that comes with Cloud EPM Platform and is available with Enterprise Performance Management Enterprise Cloud Service along with Enterprise PBCS and PBCS Plus One. You can create and edit Groovy rules in:
>
> - Planning (including these application types: Custom, Module, FreeForm, Sales Planning, Strategic Workforce Planning, and Cash Forecasting)
> - Enterprise Profitability and Cost Management
> - Financial Consolidation and Close
> - FreeForm
> - Tax Reporting

You create Groovy rules in Calculation Manager and execute them from any place that you can execute a calc script rule in an application; for example, on the Rules page, within the context of a form, in the job scheduler, in dashboards, in task lists, and so on.

Groovy rules are also supported in rulesets. You can have a combination of calc script rules and Groovy rules within a ruleset.

Groovy rules are not supported in composite forms.

You can execute jobs of type rules, rulesets, and templates synchronously from a Groovy rule.

You can write Groovy scripts to run select EPM Automate commands directly in Oracle Fusion Cloud EPM, without installing EPM Automate client on a client machine. Refer to Running Commands without Installing EPM Automate and Supported Commands in *Working with EPM Automate* for information on which EPM Automate commands can be run using Groovy and example scripts.

Oracle supports two types of Groovy rules:

- Rules that can dynamically generate calc scripts at runtime based on context other than the runtime prompts and return the calc script which is then executed against Oracle Essbase.

  For example, you could create a rule to calculate expenses for projects only for the duration (start and end dates) of the project.

Another example is a trend-based calculation that restricts the calculation to the accounts available on the form. You could use this calculation for various forms in Revenue, Expense, Balance Sheet, and Cash Flow. This allows for optimization and reuse.

- Pure Groovy rules that can, for example, perform data validations and cancel the operation if the data entered violates company policies.

Video

Your Goal	Watch This Video
Learn about training options for creating Groovy rules in Oracle Fusion Cloud Enterprise Performance Management.	 Learning Groovy in Oracle Cloud EPM

Creating a Groovy Business Rule for ASO Cubes

You can create a Groovy business rule for a ASO cube.

1. In **System View**, right-click **Rules**, then select **New**, and then enter the information in the **New Rule** dialog box.
2. Change to Script mode by clicking the arrow next to **Designer**, and then selecting **Edit Script**.

   If you select **Edit Script** on a *graphical* rule, the script designer opens and the rule is blank. Close and reopen the rule, and the rule will open again in graphical. If you save the rule in Edit Script, the rule will be empty.
3. Enter the Groovy script, and then save, validate, and deploy the rule to your application.

> **Note**
>
> Groovy business rules for ASO cubes support all variables with run time prompts (RTPs).

> **Note**
>
> When designing groovy rules, add this reference header at the top of the Groovy script:
>
> ```
> /*RTPS: */
> ```
>
> If you reference Runtime Prompts (RPTs), add the Runtime Prompts to the reference header as follows:
>
> ```
> /*RTPS: {rtp1}, {rtp2} */
> ```
>
> For example, if you had two variables named `rtpYear` and `rtpScenario`, the reference header at the top of the script would be
>
> ```
> /* RTPS:{rtpYear}, {rtpScenario} */
> ```
>
> For more information, see Class RtpValue.

Creating a Groovy Business Rule for BSO Cubes

You can create a Groovy business rule for a BSO cube.

1. In **System View**, right-click **Rules**, then select **New**, and then enter the information in the **New Rule** dialog box.
2. Change to Script mode by clicking the arrow next to **Designer**, and then selecting **Edit Script**.
3. In the top right corner of the script editor, click the arrow next to **Script Type**, and then select **Groovy Script**.
4. Enter the Groovy script, and then save, validate, and deploy the rule to your application.

> **Note**
>
> When designing groovy rules, add this reference header at the top of the Groovy script:
>
> ```
> /*RTPS: */
> ```
>
> If you reference Runtime Prompts (RPTs), add the Runtime Prompts to the reference header as follows:
>
> ```
> /*RTPS: {rtp1}, {rtp2} */
> ```
>
> For example, if you had two variables named `rtpYear` and `rtpScenario`, the reference header at the top of the script would be
>
> ```
> /* RTPS:{rtpYear}, {rtpScenario} */
> ```
>
> For more information, see Class RtpValue.

Editing the Script For a Groovy Business Rule or Template

You can edit the script for a Groovy rule or template.

To edit the script for a Groovy *rule*:

1. In **System View**, expand the **Rules** node under the application.
2. Under **Rules**, open the desired rule.
3. In the **Rule Designer**, in the third drop-down from the left, ensure that **Edit Script** is selected.
4. Edit the script as desired, and then click **Save**.

> **Note**
>
> See Editing a Business Rule in Script Mode for more information.

To edit the script for a Groovy *template*:

1. In **System View**, expand the **Templates** node under the application.

2. Under **Templates**, open the desired template.
3. In the **Template Designer**, in the third drop-down from the left, ensure that **Edit Script** is selected.
4. Edit the script as desired, and then click **Save**.

> **Note**
>
> See Creating a Groovy Template for a Planning BSO Cube for more information on creating Groovy templates.

Validating Groovy Scripts

Validate Groovy business rule scripts to ensure that your application's Groovy scripts can pass the stricter validation rules.

As you migrate to an upgraded version of Groovy, you might encounter certain issues that need to be addressed. A program called the **Groovy Script Validator** aids in finding the rules that need to be adjusted and provides a report so you can fix any issues.

> **Note**
>
> The **Groovy Script Validator** is only available only when your application contains user-created Groovy business rules.

To validate Groovy scripts:

1. Sign into your application as a Service Administrator.
2. From the Home page, select **Application**, and then **Overview**.
3. Click **Actions**, and then select **Groovy Script Validator**.

   The **Groovy Script Validator** option is only available when your application contains user-created Groovy business rules.

After the program completes verification of the Groovy scripts, you can view the Groovy validation report and fix any errors it finds. If no errors are found, no additional action is required.

To view the Groovy validation report:

1. From the Home page, select **Application**, and then **Jobs**.
2. Under **Recent Activity**, click the **Groovy validation report Job**.
3. Next to **Groovy Script Validator Report**, click  (Download File).
4. Optionally, you can download the `GroovyValidationReport.html` file from the **Inbox/Outbox Explorer** by selecting **Application**, then **Overview**, click **Actions**, and then select **Inbox/Outbox Explorer**.

> **Note**
>
> For a list of validation errors and recommended resolutions, see Resolving Groovy Business Rule Validation Issues in *Designing with Calculation Manager*.
>
> To learn more about Groovy business rules, see Creating a Groovy Business Rule in *Designing with Calculation Manager*.

Resolving Groovy Business Rule Validation Issues

As you migrate to an upgraded version of Groovy, you may encounter certain issues that need to be addressed. A program called the **Groovy Script Validator** aids in finding the rules that need to be adjusted and provides a report so you can fix any issues.

> **Note**
>
> The **Groovy Script Validator** is only available only when your application contains user-created Groovy business rules. See Validating Groovy Scripts for information on how to run the **Groovy Script Validator**.

After you run the **Groovy Script Validator** from your application, use the information in this topic to resolve validation issues.

General Guidelines

- Use Explicit Data Types instead of using `def`. For example, specify the explicit data type for variables, such as `String`, `int`, `List<String>`, or `Map<String,Integer>`.

  Note: If you are using `def` and it is not being flagged by the validation tool, you can leave it as is.

- If you get an error when working with floating numbers, use the "d" suffix to ensure the floating numbers are treated as "double" instead of "BigDecimal". For example, `it.data = 1212121212.111d`.

- If you get an error that you do not understand, verify that the first line of the script looks like this: `/*RTPS: */`

  Even if you have no RTPs, make sure this is the first line.

  If you have RTPs, add them to the first line like this: `/*RTPS: {selectedDepartment} {hireDate} {newEmployeeName} */`.

- If you get an error with "Unexpected input", ensure that all your List declarations and casting do not have an empty type such as `List<>`.

  Examples:

  – Change `List<>` to `List` or `List<Object>`.

  – Change `targetGrid.addRow((List< String>) listRow[0] , (List<>) listRow[1])` to `targetGrid.addRow((List< String>) listRow[0] , (List) listRow[1])`

> **Note**
>
> Starting in the October update (25.10), you will no longer be required to replace deprecated Date functions with corresponding Calendar functions as part of the validation process. This will be resolved and will no longer be an issue. The Date.format(), Date.getAt(), Date.parse() etc. methods will work and will no longer show validation errors.

> **Note**
>
> If you encounter validation errors caused by templates or rules that are not intended to be launched from Planning, remove these items by navigating to the Deployment View in Calculation Manager. Under the "To be Deployed" node, uncheck the relevant rules and templates. Then, redeploy all artifacts by right-clicking the application node and selecting "Deploy".

Errors and Suggested Fixes

Table 3-2 Errors and Suggested Fixes

Error	Example Code	Possible Cause	Possible Solution
No such property: `<groovy variable name>` for class: `groovy.lang.Binding`	N/A	This may be the result of not using the RTPS model, and the variables are referenced via square brackets or curly braces. This results in the Groovy engine trying to evaluate them as Groovy expressions.	Use the `rtps.<variable name>`, or recheck the expressions within the braces or brackets.

Table 3-2 (Cont.) Errors and Suggested Fixes

Error	Example Code	Possible Cause	Possible Solution

Cannot assign value of type `type java.lang.Object` to variable of type `double`	`def data = []`
	`double valFor = data[0]`
		The error occurs because the List data is not explicitly typed, so Groovy treats it as a `List<Object>`. When you access an element from the List using `data[0]`, it returns an object, which cannot be directly assigned to a `double` variable.	To resolve this issue, do one of the following:
			• Explicitly type the list .
			Define the list with the correct type, such as `List<Double>`.
			For example:
			`List<Double> data = []`
			`double valFor010 = data[0]`
			• Cast the value to `double`.
			Use the `as double` syntax to explicitly cast the object to a double value.
			For example:
			`def data = []`
			`double valFor010 = data[0] as double`

Cannot assign value of type `java.lang.Object` to variable of type `int`	`def finalList = mergeList.get(i)`
	`int pmntfrequency = finalList.get(3)`
		The error occurs because the object being assigned is not explicitly typed or cast to an `int`. In this case, `finalList.get(3)` returns an object, which cannot be directly assigned to an `int` variable.	To resolve this issue, cast the value to `int`.
			Use the `as int` syntax to explicitly cast the object to a `int` value or use the `as int` tag.
			For example:
			`def finalList = mergeList.get(i)`
			`int pmntfrequency = (int) finalList.get(3)`
			`int pmntfrequency = finalList.get(3) as int`

Table 3-2 (Cont.) Errors and Suggested Fixes

Error	Example Code	Possible Cause	Possible Solution
Cannot assign value of type `java.util.List <java.lang.String>` to variable of type `java.lang.String[]`	`String[] arrGridMbrs = it.getMemberNames()`	The error occurs because `it.getMemberNames()` returns a `List<String>`, which is not directly assignable to a `String[]` array.	To resolve this issue, add the `as String[]` cast to explicitly convert the `List<String>` to a `String[]` array. For example: `String[] arrGridMbrs = it.getMemberNames() as String[]`

Cannot call the following:
`oracle.epm.api.grid.DataGridDefinitionBuilder#addPov(java.util.List <java.lang.String>, java.util.List <java.util.List>)`
with agruments
`[java.util.List <java.lang.Object>, java.util.List <java.lang.Object>]`
	`def columnDims = []`
	`def columnMbrs = []`
	`builder.addColumn(columnDims, columnMbrs)`
		The error occurs because `List<String>` and `List<List<String>>` arguments are passed in as `List<Object>` and `List<Object>`.	To resolve this issue, specify the data type of the variable.
			For example:
			`List<String> columnDims = []`
			`List<List<String>> columnMbrs = []`
			`builder.addColumn(columnDims, columnMbrs)`

Cannot call the following:
`oracle.epm.api.grid.DataGridDefinitionBuilder#addRow(java.util.List <java.lang.String>, java.util.List <java.util.List>)`
with arguments
`[java.util.List <java.lang.Object>, java.util.List <java.lang.Object>]`
	`dataGridDefinitionBuilder.addRow(['Account', 'Period'], [['Account1'], ['P1']])`	The error occurs because `List<String>` and `List<List<String>>` arguments are passed in as `List<Object>` and `List<Object>`.	To resolve this issue, add the `as String[]` cast to explicitly convert the `List<String>` to a `String[]` array.
			For example:
			`dataGridDefinitionBuilder.addRow(['Account', 'Period'] as List<String>, [['Account1'], ['P1']] as List<List<String>>)`

Table 3-2 (Cont.) Errors and Suggested Fixes

Error	Example Code	Possible Cause	Possible Solution

Cannot call the following:
`oracle.epm.api.grid.DataGridDefinitionBuilder#addColumn(java.util.List<java.lang.String>, java.util.List<java.util.List>)`
with arguments
`[java.util.List<java.lang.Object>, java.util.List<java.lang.Object>]`
	`dataGridDefinitionBuilder.addColumn(['Account', 'Period'], [['OCX_Payment Frequency'], ['Begbalance']])`	The error occurs because `List<String>` and `List<List<String>>` arguments are passed in as `List<Object>` and `List<Object>`.	To resolve this issue, add the `as String[]` cast to explicitly convert the `List<String>` to a `String[]` array
			For example:
			`dataGridDefinitionBuilder.addRow(['Account', 'Period'] as List<String>, [['Account1'], ['P1']] as List<List<String>>)`

Cannot find matching method `java.util.Date#getAt(int)`	`rtps.endDate.getDataAsDate().getAt(Calendar.YEAR)`	The error occurs because the `getAt(int)` method, which was previously available in earlier versions of Groovy, is no longer available.	To resolve this issue, enter the following:
			`Calendar calendar = Calendar.getInstance()`
			`calendar.setTime(rtps.endDate.getDataAsDate())`
			`int year = calendar.get(Calendar.YEAR)`

Cannot find matching method `java.util.Date#format(java.lang.String)`	`rtps.endDate.getDataAsDate().format("yyyy-MM-dd")`	The error occurs because the format method, which was previously available in earlier versions of Groovy, is no longer available.	To resolve this issue, enter the following:
			`SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd")`
			`String formattedDate = sdf.format(rtps.endDate.getDataAsDate())`

Table 3-2 (Cont.) Errors and Suggested Fixes

Error	Example Code	Possible Cause	Possible Solution

Cannot find matching method `oracle.epm.api.model.Application#getDimension(java.lang.String, java.util.List)`	`List <Cube> cube = operation.application.getCubes() Dimension dimScenario = operation.application.getDimension("Scenario", cube.toArray(new Cube[cube.size()]))`	A `List<Cube>` is being passed into the second argument which expects and 0 or more cubes.	To resolve this issue, do one of he following:
			• Ensure the declared type is correct and he method exists.
			• Enter the following:
			`List <Cube> cubes = operation.application.getCubes()`
			`Dimension dimScenario = operation.application.getDimension("Scenario", cubes.toArray(new Cube[cube.size()]))`
			• If only a dimension valid for a particular cube is needed, enter the following:
			`Cube cube = application.getCube("Plan1")`
			`Dimension dimScenario = operation.application.getDimension("Scenario", cube)`

Java API Reference for Groovy Rules

For Enterprise applications, a Java API Reference is available to use as you create Groovy rules.

The Java API Reference includes examples that demonstrate the syntax and power of the EPM Groovy object model.

To view the Java API Reference, see the Java API Reference for Oracle Enterprise Performance Management Cloud Groovy Rules on the Cloud Help Center. You can also

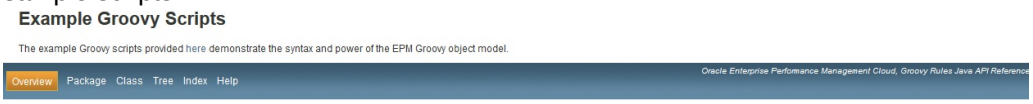
access this reference from the Oracle Enterprise Planning and Budgeting Cloud Academy. To access the Academy, sign in, and then click **Academy**.

Groovy Business Rule Examples

Example Groovy scripts are available.

To see example Groovy scripts:

1. See the Java API Reference for Oracle Enterprise Performance Management Cloud Groovy Rules, https://docs.oracle.com/cloud/latest/epm-common/GROOV/.
2. Do one of the following:
   - Under **Example Groovy Scripts** on the main page, click the word "here" to view sample scripts:

   - Under **All Classes** in the left pane, click a class to see the examples for that class.

     For example, to see Strategic Modeling examples, click the StrategicModel class in the left pane.

Groovy Business Rule Tutorial Videos

Watch these tutorial videos for details and best practices when implementing and using Groovy Business Rules.

Your Goal	Watch this Video
See the training options for creating Groovy rules in Oracle Fusion Cloud Enterprise Performance Management:	Learning Groovy in Oracle Cloud EPM
Use Groovy rules to calculate incrementally loaded data in Data Management.	Calculating Incrementally Loaded Data in Data Management Using Groovy Rules
Use Groovy templates to improve usability and calculation performance for user actions.	Customizing Actions to Improve Performance Using Groovy Templates

Groovy Business Rule Tutorials

Complete these tutorials for hands-on examples on implementing Groovy Business Rules.

Implement Groovy: Basic Tasks

Your Goal	Learn How
Learn about the Groovy scripting language and how to create a Groovy script for your business process in Oracle Fusion Cloud Enterprise Performance Management.	Introduction to Groovy Business Rules
Implement Groovy scripts to work with data grids and data grid iterators. • Use conditional logic to set the background colors in a data grid • Implement performance benchmarking • Set default data values in a grid	Working with Data Grids and Iterators in Groovy
Create a Groovy script to work with business process metadata, in this case, moving a member from one parent to another. The script includes RTPs to prompt users for input. You'll also learn how to create a right-click action menu with a menu item to call the script, and how to associate the action menu with a data form.	Moving Dimension Members with Groovy
Push data from a source location to a target location using Groovy rules and Smart Push.	Moving Modified Data Using Groovy and Smart Push

Implement Groovy: Advanced Tasks

Your Goal	Learn How
Implement a Groovy script that validates data entry against allowed ranges stored in a driver cube.	Validating Data Entry Rules with Groovy

Your Goal	Learn How
Create a Groovy script to work with business process metadata, in this case, adding a member to a dimension. The script includes RTPs to prompt users for input. You'll also learn how to create a right-click action menu with a menu item to call the script, and how to associate the action menu with a data form.	Adding Dimension Members with Groovy
Generate focused calculation scripts in your business process to calculate only data that has been edited, instead of the entire data entry form.	Calculating Modified Data Using Groovy
Understand Strategic Modeling concepts and learn how to integrate data between strategic models and business process applications using Groovy scripts.	Integrating Data between Planning and Strategic Models Using Groovy
Create a Groovy template with interactive Design Time Prompts (DTPs).	Creating Groovy Templates
Learn how to use a Groovy business rule in the business process to export dimension metadata from Cloud EPM, and then import the metadata into the business process.	Integrating Dimension Metadata from Enterprise Data Management Cloud with Planning Using Groovy Rules

Implement Groovy: REST API

Your Goal	Learn How
Call a Data Management REST API to execute a data load rule that will load the latest product volumes for the user's entity into the business process.	Calling an Internal REST API Using Groovy

Your Goal	Learn How
Call an external Rest API from a Groovy script in your business process. You'll also learn how to create a right-click action menu with a menu item to call the script, and how to associate the action menu with a data form.	Calling an External REST API Using Groovy

Opening a Business Rule

You open a business rule from the System View that is displayed by default when you open Calculation Manager.

You can also open a rule from within the Custom View, the Filter View, or the Deployment View.

To open a business rule, do one of the following:

- Right-click the rule, and then select **Open**.
- Double-click the rule.

> **Note**
>
> In Oracle Financials Cloud, if the rule is locked by another administrator, you will get a message stating: "objectname opened in read only mode. The object is being currently edited by: FINUSER2". If this happens, complete the following steps:
>
> 1. Click **OK**.
> 2. Right-click on the rule or ruleset and select **Unlock**. The following message is displayed:
>
>   Do you want to unlock the selected item? Any changes by the user who is editing it will be lost.
> 3. Click **OK** to open the rule or **Cancel** to close the rule without opening it.
>
>   If the rule is unlocked, the administrator who was editing the rule does not get a message. If the rule is edited by the original administrator and saved, and then the administrator who unlocked the rule makes a change and saves, then the last save is kept, and the edits made by the original administrator are overwritten.

Editing Business Rules

Related Topics

- Editing a Business Rule
  You can edit the structure of a business rule by adding to, removing, or changing its components (including formulas, scripts, conditions, ranges, and loops).
- Editing a Business Rule in Script Mode
  You can edit the script of a business rule.

- Options Available When Editing in Script Mode

Editing a Business Rule

You can edit the structure of a business rule by adding to, removing, or changing its components (including formulas, scripts, conditions, ranges, and loops).

You can also edit the properties of the business rule's components and the properties of the business rule itself.

You can edit the following business rule properties:

- Name and caption
- Description and comments
- Dimensions, members, and variables

To edit a business rule:

1. Open the rule.
2. In the Rule Designer, add new components, and copy and delete existing components from the rule's flow chart.
   - To add a component, drag an object from **New Objects** or **Existing Objects** and drop it into the flow chart.

     When you add an existing formula or script component to the flow chart, by default, the formula or script becomes a shared object. If you do not want it to be shared, clear the **Shared** check box in the formula or script component's **Properties**. See Sharing Script and Formula Components.
   - To copy a component, right-click the component, then select **Copy**, and then paste it into the flow chart.
   - To delete a component, right-click the component, and then select **Remove**.
3. In **Properties**, edit properties of the rule.

   The properties change as you add components to the rule and move among the components in the flow chart. To enter properties for a specific component, select the component in the flow chart. See Using Components to Design Business Rules and Templates.)
4. Click  to save the changes.

> **Tip**
>
> As you edit the components in a business rule, you can increase or decrease the size of the component icons and the amount of detail that is displayed in the flow chart. To do this, use  to zoom in and out within the flow chart.
>
> When the flow chart is displayed in a small size, the component captions do not display, but you can place your mouse pointer over any icon to read its caption. Regardless of the size of the components in the flow chart, you can select a component to view its properties.

Editing a Business Rule in Script Mode

You can edit the script of a business rule.

By default, you create a business rule in graphical mode using the Rule Designer to design the graphical flow of the rule. After you create and save a business rule, you can edit it in graphical mode or script mode. If you choose to edit it in script mode, you can return to editing in graphical mode later.

To edit a business rule in script mode:

1. Open the rule.
2. In the Rule Designer, in the drop-down next to **Designer**, select **Edit Script**.

   To switch back to graphical mode, in the drop-down next to **Edit Script**, select **Designer**.

   When you switch from editing in graphical mode to script mode, if the business rule contains shared components (such as formula and script components) or templates, the script of the business rule contains only references to the shared components and templates in the Script Editor. The complete generated script of the shared components and templates is displayed on the **Script** tab in the bottom pane.

   Note the following:

   - When you select **Edit Script** in an ASO rule, the script is automatically converted to Groovy. If you then save the rule, the script is saved in Groovy and cannot be converted back to graphical. To preserve the graphical formatting, select **Save As** and save the rule with the Groovy script with a different name.
   - When you select Edit Script in a BSO rule, the graphical rule is converted into script mode that is based on calculation script syntax. To convert the script to Groovy, selecting **Groovy Script** as the Script Type.
   - Do not use the following keywords when editing in script mode:
     - `/*STARTCOMPONENT*/`
     - `/*STARTCOMPONENT:SCRIPT*/`
     - `/*STARTCOMPONENT:FORMULA*/`
     - `/*ENDCOMPONENT*/`
     - `/*STARTRULE*/`
     - `/*ENDRULE*/`
     - `/*STARTTEMPLATE*/`
     - `/*ENDTEMPLATE*/`
3. Edit the script as desired.

   See Options Available When Editing in Script Mode.
4. Click  .

Review the following for additional information:

- Options Available When Editing in Script Mode
- Reviewing Scripts

Options Available When Editing in Script Mode

Table 3-3 Options Available When Editing in Script Mode

Icon	Description	See Also
	Display or hide the script line numbers. Line numbers are displayed by default.	
	Insert a function and its parameters Auto suggest is provided for functions. When you type the first few characters of the function, press CTRL+Space to display the suggestions. Select one of the suggestions, and press Enter to complete the function.	Working with Functions
	Insert cross-dimensional members	Adding Members and Functions to a Component
	Insert a range of members selected from dimensions	Adding Members and Functions to a Component
	Insert a variable	Working with Variables
	Insert a Smart List	Working with Smart Lists
	Edit a template	Working with System Templates
	Comment	
	Uncomment	
	Show/hide comment	
	Format code When you Format Code, the lines of the script are grouped together and are indented, and blank lines are removed.	
	Verify syntax	
	Wrap the script so that long lines of the script that scroll off the page display on multiple lines on the same page	
 	Code completion on / Code completion off Enables or suppresses suggestions for completing the code For example, to initiate code completion, after you type `FIX`, type SHIFT + Space so Calculation Manager enters () `ENDFIX` for you.	
	Find and replace a text string in the script	Searching for a Text String in a Business Rule Script
	Select whether to **Match Case** or **Match Whole Word** when searching for a text string	Searching for a Text String in a Business Rule Script

Saving Business Rules

Save business rules to the application and application type for which they were created.

Related Topics

- Saving a Business Rule
- Saving a Business Rule with a Different Name

Saving a Business Rule

When you save a business rule, it is saved to the application and application type for which you created it. After you save a business rule, you can deploy, validate, and launch it. You can deploy and validate the rule in Calculation Manager, and you can launch the rule from Planning.

To save a business rule after you create or edit it, click  .

> **Note**
>
> To see the business rule in the **System View** after you save it, you may need to refresh the application list. To do this, right-click any node or object in **System View**, and then select **Refresh.**

Saving a Business Rule with a Different Name

You can save a business rule with a different name using Save As. You can also copy a rule from one ruleset to another within the same ruleset type using Save As. Save As creates a copy of the original business rule with a different name to distinguish it from the original.

To save a business rule with a different name:

1. Double-click the rule.
2. In the Rule Designer, select **Actions**, and then **Save As**.
3. In **Save As**, enter the rule's new name, and select the **Application**.
4. Select the **Plan Type**.
5. Click **OK**.

   The new rule is added to the application list in **System View**.

Running a Business Rule

You must open a business rule before you run it.

To run a business rule:

1. Double-click the rule, or right-click the rule, and then select **Open**.
2. When the rule opens click  (Launch button).

Copying Business Rules

You can copy a business rule to another application and plan type.

When you copy a business rule to another application, you create a new business rule with another name. Any variables used in the business rule are also copied to the new application.

To copy a business rule to another application and plan type:

1. In **System View**, right-click a rule, and then select **Copy To**.
2. In **Copy To**, enter a new name for the business rule or accept the default name, and select an application, application type, and plan type.

   You cannot copy a business rule to more than one application and plan type.
3. If you copy a business rule with shared components to another application, and you want to copy the shared components to that application, select **Copy Shared Components**.

   This creates the shared components in the copied application, and the copied rule uses the shared components from its application and plan type. If you do not select **Copy Shared Components**, the shared components are copied in the rule, but the copied rule uses the shared components from the original rule's application and plan type.

   When you copy a business rule that has variables to another application, the variables are created in the copied to application. If the variables already exist in the copied to application, the variables are created as rule level variables when copied.
4. If you copy shared components, select **Replace Existing Objects** to overwrite objects with the same name.

Searching in Business Rules

Search for a text string in a business rule script. Search for and replace text in a graphical business rule.

Related Topics

- Searching for a Text String in a Business Rule Script
- Searching for and Replacing Text in a Graphical Business Rule

Searching for a Text String in a Business Rule Script

To search for a text string in a business rule script:

1. Open a business rule that contains script.
2. In the Rule Designer, select the **Script** tab.
3. Enter the text for to search, and then click  .

Searching for and Replacing Text in a Graphical Business Rule

When you search for a string, Calculation Manager starts the search with the first component after the component selected in the flow chart, continues through to the end of the flow chart, and starts over at the beginning of the flow chart until it reaches the component selected in the flow chart.

After one occurrence of the text string is found, if you want to search for another, you must start the search again to find the next occurrence.

To search for and replace text in a graphical business rule:

1. Open the business rule.
2. In the Rule Designer, right-click the **Begin** or **End** icon in the business rule, or a component within the business rule's flow chart, and then do one of the following
   - Select **Find** to find instances of the text string.
   - Select **Replace All** to find and replace instances of the text string.

Printing a Business Rule

You can print a business rule's properties, its flow chart, and the details of its components.

For example, if you print a business rule that contains a formula component for allocation expenses, the print out shows the formula syntax, the functions and variables that comprise the formula, a summary of the steps in the rule's flow chart (not in graphical form), and the rule's properties.

If you print a business rule that is in script only:

- The Rule Details section in the PDF file contains the business rule script.
- The Flow Chart section oin he PDF file contains only a Begin and an End node.
- The Flow Summary section in the PDF file is empty.

> **Note**
>
> You cannot print business rulesets or components. You can print components if they are used in a business rule that you print.

To print a business rule:

1. In **System View**, right-click a business rule, and then select **Print**.
2. Enter the information in the **Print Preview** dialog box, and then click **Generate PDF.**

   A PDF file of the business rule is opened in Adobe Acrobat.

Deleting a Business Rule

You can delete a business rule only if it is not used by other rules or rulesets.

If the rule is being used, you must remove the business rule from the rules and rulesets using it, or make copies of it for the rules and rulesets using it before you delete it.

To delete a business rule:

1. In **System View**, right-click the rule, and then select **Show Usages** to ensure that it is not being used by other rules or rulesets.
2. If the rule is not being used by other rules and rulesets, right-click the rule, and then select **Delete**.
3. Click **OK** to confirm deletion of the rule.

Defining Common Dimensions In Business Rule Components

Define common dimensions by opening the business rule and selecting the members, variables, and functions that are common for each dimension.

To define the common dimensions in business rule components:

1. Open a business rule.
2. In the Rule Designer, click **Begin** or **End** in the flow chart.
3. On the **Global Range** tab, select a dimension, then click  , and then select the members, variables, and functions that are common for each dimension.

   The values you select for the dimensions are the values that are calculated when the rule is launched.

   If you select a variable, select **Link Variable Dynamically** to dynamically link the variable to the rule so that when changes are made to the variable, the changes are updated in the rule.

> **Note**
>
> Select **Exclude Grid Values** to create an "Exclude/EndExclude" script in the business rule instead of a "Fix/EndFix" script.

Analyzing and Debugging Business Rules

Analyze a business rule by running the rule and capturing statistical information. Debug a business rule by running the rule and examining its script.

Related Topics

- Analyzing Business Rules
- Analyzing the Script of a Business Rule
- Comparing the Scripts of Business Rules
- Comparing a Changed Business Rule to a Saved Business Rule
- Debugging Business Rules
- Disabling a Component in a Business Rule

Analyzing Business Rules

When you analyze a business rule, you run the business rule and capture statistical information such as how long a component took to execute, how many times it was run, and the values of a member intersection before and after the component was run. The amount of time the analysis takes depends on the memory of your system.

> **Note**
>
> The Analyze feature is not available for Member Range, Member Block, Fixed Loop, or Condition components.

To analyze a business rule:

1. Open a business rule.
2. In the Rule Designer, select **Actions**, and then **Analyze**.
3. In the **Enter Analysis Criteria** dialog box:
   - Enter the number entries to display in the analysis, or select **Show All** to display entries.

     The number entered here is the number of member intersections to calculate and display. For example, assume you have these dimensions in the application:

     – Year = 12 members
     – Measures = 50, 525 members
     – Product = 450 members
     – Market = 2000 members
     – Scenario = 4 members

     In this example, the number of member intersections to calculate is: 2, 182, 680, 000 (12 x 50,525 x 450 x 2,000 x 4 or Year x Measures x Product x Market x Scenario)
   - For each dimension, select the members to analyze.

     The following options are available to select members:

     – Click  to access the Member Selector dialog box.
     – Select a dimension, and then click  .
     – Type in the member.

     You can only select one member per dimension.
4. Click **OK** to start the analysis.
5. When the analysis is complete, click a component in the business rule flow chart, and then select the **Analysis Information** tab.

> **Tip**
>
> To export the analysis information, right-click in the grid in the **Analysis Information** tab, and then select **Export**.

> **Note**
>
> The analysis may not show the correct begin and end values because of Oracle Essbase optimizations. If the analysis does not show a begin and end value for at least one intersection, the count and the elapsed time might not be correct.

Hiding Members of Tracked Dimensions

To hide the members of the dimensions that are being tracked, in the **Analysis Information** tab, select **Hide dimensions(s) that are being tracked**. If you do not select this option, all the members processed for the dimension are displayed.

The before and after values displayed in the **Analysis Information** tab indicate which member changed the tracking member values. The before and after values are the values for the members you are tracking. For example, if you are tracking Product 200-30, you may see lines for Product 200. The values shown on the line for Product 200 are the values for 200-30.

Capturing Statistical Information

To capture statistical information when analyzing a business rule, keep in mind the following points:

- The internal call to Oracle Essbase must be surrounded by a member block.
- If you leave only one dimension empty in the Fix statement, and you do not use a tracking member from that dimension, then that dimension or one of its descendants is used for the member block.
- If you leave only one dimension empty in the Fix statement, and you specify a tracking member from that dimension that is a dynamic calculation member, then the member block is that dimension or one of its descendants that is not a dynamic calculation or label only member.
- The **Analysis Information** tab displays values for the tracking member, and the member used in the member block.

Analyzing the Script of a Business Rule

You can run a graphical business rule or a script business rule and analyze its script.

To analyze the script of a business rule:

1. In **System View**, right-click a business rule, and then select **Open**.
2. With the rule displayed, select **Actions**, and then **Analyze Script**.

   This runs the business rule.
3. If the business rule contains runtime prompt variables, in the **Enter RTP Values** dialog box, enter runtime prompt values, then specify whether to apply the values to the business rule, and then click **OK**.
4. In the **Script Analysis** tab, analyze the script.

   If a line in the script has analysis information,  is displayed next to the line. Select a line to review its analysis information in the **Analysis Information** area on the right of the screen. If there is no analysis information for a line, "Analysis information is not available" is displayed. The line that has the longest execution time is highlighted in red.

The **Properties** area displays information about the business rule, such as its name, application, plan type, and the length of time (in milliseconds) that it took to analyze the script.

5. When you finish reviewing script information, click  to close the **Script Analysis** tab.

Comparing the Scripts of Business Rules

You can compare the scripts of business rules to see how they are different from each other, to see the changes made, or to see the scripts of graphical business rules.

> **Note**
>
> You cannot compare more than two business rules at a time. In addition, you cannot compare versions of a business rule. For example, you cannot compare a previous version of a business rule to the version that you just saved.

To compare the scripts of two business rules:

1. In **System View**, select two business rules.
2. Right-click the selected business rules, and then select **Compare Script**.

   In the **Compare Scripts** window:

   - If an addition was made to either script, the line is displayed with a light gray background.
   - If a deletion was made from either script, the line is displayed with a dark gray background.
   - If either script was changed, the line is displayed with a yellow background.
   - The first column displays the line number of the script.
   - The second column displays one of three characters:
     - * indicates that the line in the script displayed in the left column is different from the line in the script displayed in the right column.
     - + indicates that a new line was added to the script displayed in the right column when compared to the script displayed in the left column.
     - – indicates that a line was deleted from the script displayed in the right column when compared to the script displayed in the left column.
   - The third and fourth columns display the two scripts.
   - The bottom part of the window is divided into two panes that display the complete lines of the scripts you selected in the top panes. The first pane displays the script in the left column, and the second pane displays the script in the right column.
3. Click **OK** when you finish comparing the scripts.

Comparing a Changed Business Rule to a Saved Business Rule

You can view the scripts of the saved version of a business rule and a changed version of the business rule to see how the two versions are different from each other, to see changes you made, or to see the scripts of a saved and changed graphical business rule side-by-side.

To compare a saved and a changed business rule:

1. Open a graphical or calculation script business rule.
2. Select **Edit**, and then **Compare with saved version**.

   In the **Compare Scripts** window:

   - If an addition was made to the business rule or calculation script, the line is displayed with a light gray background.
   - If a deletion was made from the business rule or calculation script, the line is displayed with a dark gray background.
   - If the business rule or calculation script was changed, the line is displayed with a yellow background.
   - The first column displays the line number of the business rule or calculation script.
   - The second column displays one of three characters:
     - * indicates that the line in the business rule or calculation script displayed in the left column is different from the line in the business rule or calc script displayed in the right column.
     - + indicates a new line was added to the business rule or calculation script displayed in the right column when compared to the business rule or calc script displayed in the left column.
     - – indicates a line was deleted from the business rule or calculation script displayed in the right column when compared to the business rule or calc script displayed in the left column.
   - The third and fourth columns display the saved and changed business rule or calculation script.
   - The bottom part of the window is divided into two panes that display the complete lines of the business rule or calculation script you selected in the top panes. The first pane displays the saved version of the business rule or calculation script in the left column, and the second pane displays the modified version of the business rule or calculation script in the right column. Select a row in the script to display it in the bottom pane. The saved version is on top, and the modified version is on the bottom.
3. Click **OK**.

Debugging Business Rules

Debugging a business rule involves running the business rule and examining its script line by line to see how the script executes.

You can debug business rules for the following applications:

- Oracle Essbase block storage applications, if you have write privileges
- Planning applications, if you have launch privileges

> **Note**
>
> You cannot edit a business rule while you are debugging it.

To debug a business rule:

1. Open a business rule.
2. In the Rule Designer, select **Actions**, and then **Debug**.

   The script of the business rule is displayed in the script debugger. Each statement in the script is displayed on a separate line.
3. Debug the statements in the business rule script.

   As you debug statements, you can:

   - Insert and remove break points where you want to stop the execution of the script to examine the values of the intersections of members in the statement. When the execution stops at a break point, the values of the intersection of all members in the statement are displayed.

     To add a break point, right-click to the right of the break point, and then select **Add Break Point**. You can add break points to assignment and conditional statements. You can add a break point only in lines of the script that show the break point grayed out. To remove a break point, right-click it, and then select **Remove Break Point**.
   - Add a condition to a break point to stop the execution of the statement only if the condition is met. Only members used in the statement with the breakpoint can be used in the condition.

     To add a condition to a break point, right-click to the right of the break point, and then select **Add Condition**. In the **Add Condition** dialog box, click  , and then enter the condition in the **Condition Builder**.

     To edit a condition, right-click it, and then select **Edit Condition**.
   - Debug statements with break points.

     To debug a statement with a break point, right-click the statement, and then select **Start Debug**. The statement you are debugging is highlighted. The members of the statement, the break points, and the values of the intersection of the members before and after execution are displayed in the following tabs:

     – **Members**—Shows the current intersection of members at the debug breakpoint. To see the next intersection of members, click **Resume Debugging**.

     – **Breakpoints**—Shows the expressions from the script that contain breakpoints. The **Values at the Break Point** tab displays the expression members with the values as they were before and after the debug was run.

> **Tip**
>
> To resume debugging, click  . You must continue clicking  until all of the member intersections are debugged. After all of the member intersections are debugged, the message, "The script debugging is complete" is displayed.

Disabling a Component in a Business Rule

You can disable a component in a business rule. For example, you may want to exclude a component from validation to analyze which components are causing validation errors.

To disable a component in a business rule:

1. Select the component in the business rule flow chart.
2. In **Properties**, select **Disabled**.

After you disable a component, and save the business rule to which it belongs, the component icon is disabled in the business rule flow chart. Any component that you move to, or create within, the disabled component inherits the Disabled status of the disabled component and is not calculated as part of the business rule.

> **Note**
>
> You cannot disable a Condition component. If you disable a component that is shared, it is disabled only for the business rule in which it is disabled.

Refreshing Business Rules or Business Rulesets

In the System View, Custom View, and Deployment View, you can refresh any level of the application list.

You can refresh the Planning application type, a Planning application, a plan type, multiple rulesets or rules, or one ruleset or rule.

By default, when you refresh any node in the application list, all of the rules, rulesets, components, and templates belonging to it are refreshed. However, refreshing the rulesets or rules within an application does not refresh higher levels in the application list or rulesets or rules that belong to other applications.

For example, if you refresh a rule within a Planning application and plan type, all rules within that application and plan type are refreshed, but no rules within other plan types or other Planning applications are refreshed.

To refresh a business rule or ruleset, right-click the rule or ruleset, and then select **Refresh**

> **Note**
>
> To update the metadata changes from Planning, right-click the application node in System view and select **Refresh**. You can also right-click the Planning application type, the application, or the plan type that contains the business rules you want to refresh, and select **Refresh**.

Refreshing Metadata Changes

To refresh the metadata changes (cubes, dimensions, Smart Lists, UDAs) from Planning:

1. In System view, right-click the application node or the cube node.
2. Select **Refresh**.

Showing the Usages of a Business Rule or Ruleset

Display the rules, templates, and rulesets that are using a business rule or business ruleset.

Viewing the usages of a rule or ruleset is useful when you want to delete the rule or ruleset and need to know what objects are using it.

To show the usages of a business rule or ruleset:

1. Right-click the rule or ruleset, and then select **Show Usages**.
2. Review the information in the **Usages** dialog box, and then click **OK**.

Optimizing Business Rules

Leverage available tools and effectively manage your rules.

Related Topics

- Overview of Business Rule Optimization
- Steps to Optimize Business Rules
- Identifying Slow Running Business Rules in the Planning Application Activity Report
- Identifying Slow Running Business Rules in Calculation Manager Log Messages
- Using Log Messages to Optimize Business Rules
- Example Business Rule

Overview of Business Rule Optimization

You can diagnose performance issues with business rules and pinpoint and address any issues using Planning activity reports. Activity reports identify which business rules are taking the longest to execute. You can then open the rules in Calculation Manager, review the log, and optimize the steps within the rule to improve performance.

Steps to Optimize Business Rules

To optimize business rules:

1. From the Planning Application Activity Report, find out which business rules are taking the longest time to execute (see Viewing Activity Reports in *Administering Planning*).
2. Clone the application from the production service instance to the test service instance.
3. For each slow-running business rule, complete the following steps:
   - Run the business rule from Calculation Manager (see Running a Business Rule).
   - Use the Log Messages in Calculation Manager to identify what steps in the rule are taking the longest time.
   - Optimize the identified steps and run the business rule again to ensure that it takes less time to execute but still calculates the correct results.
4. Once the business rules are optimized, migrate the changes back to the production service instance.

Identifying Slow Running Business Rules in the Planning Application Activity Report

Note the business rule/calculation script name in the reports.

Top 5 Worst Performing Calc Scripts Commands over 1 Min

Duration (Min:Sec)	Begin Time	End Time	Context	Calc Script Command
01:20	02:46:18	02:47:38	Calc Script: YTD_Calc Blocks Read: 3,937 Blocks Updated: 3,339 Blocks Created: 3,190	FIX ("BaseData", "Plan", "FY17", "FY16") .... See More
01:04	02:57:26	02:58:30	Calc Script: YTD_Calc Blocks Read: 3,321 Blocks Updated: 2,731 Blocks Created: 2,590	FIX ("BaseData", "Plan", "FY17", "FY16") .... See More

Top 10 Worst Performing Calc Scripts

Duration (Min:Sec)	Begin Time	End Time	Application	Calc Script
01:20	02:46:18	02:47:38	Vision	YTD_Calc
01:04	02:57:26	02:58:30	Vision	YTD_Calc

Top 10 Worst Performing Business Rules over 30 Seconds

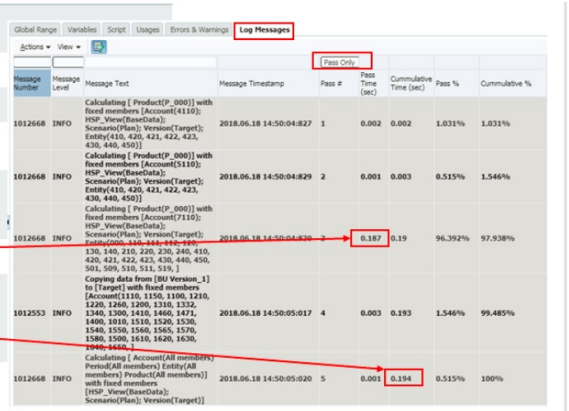
Duration (Min:Sec)	Begin Time	End Time	User	Business Rule	Run Time Prompts
01:20	02:46:18	02:47:38	epm_default_cloud_admin	YTD_Calc	
01:04	02:57:26	02:58:30	epm_default_cloud_admin	YTD_Calc	

> **Note**
>
> See "Viewing Activity Reports" in *Administering Planning* for more information on Planning activity reports.

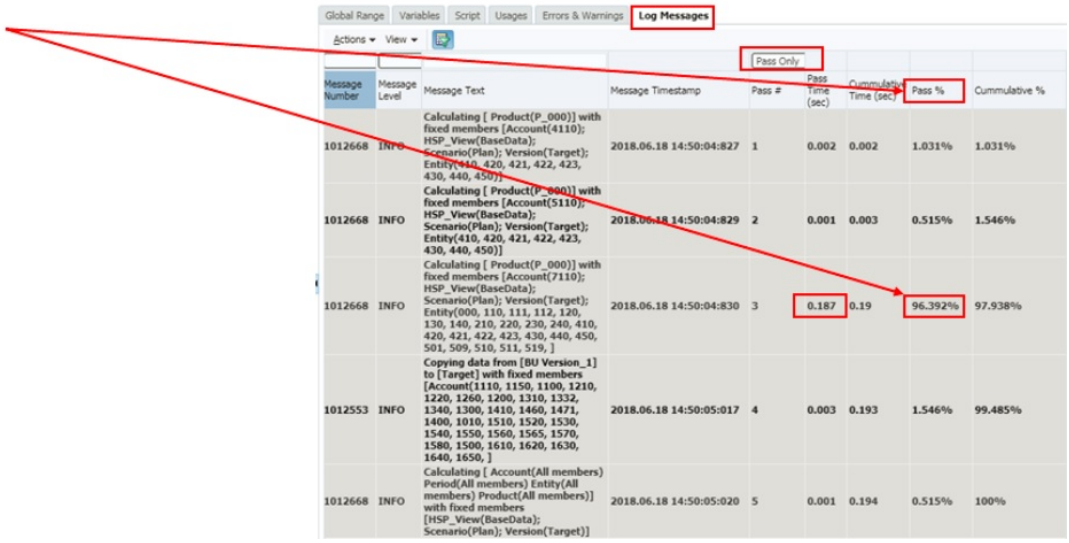
Identifying Slow Running Business Rules in Calculation Manager Log Messages

- After the business rule is executed, a new Log Messages tab is displayed.:

Message Number	Message ID of the log entry
Message Level	Level of the message (e.g., INFO)
Message Text	Complete text of the log entry
Message Timestamp	Timestamp when the log entry was generated
Pass #	Number of the current pass. The highest number in this column is the total number of passes in the business rule.
Pass Time (sec)	Execution time, in seconds, of the current pass only
Cumulative Time (sec)	Time in, in seconds, since the start of the rule ~ (i.e., previous Cumulative Time + current Pass Time) The last entry in this column is the total business rule execution time
Pass %	Pass time as a percentage of the total business rule elapsed time
Cumulative %	Percentage of the total time of all passes in the business rule to that line When all passes are complete, this should show 100%.



- Using this information, you can see which pass (or passes) have taken the highest percentage of the total calculation time to run.



Using Log Messages to Optimize Business Rules

- The Log Messages tab helps the business rule implementer determine how the total business rule execution time is spread across the number of calculation passes.
- The start of a new pass is shown in the Log Messages with a bold line.
- View the **Pass #** and **Pass %** columns to determine which passes consume most of the total business rule execution time.
- The sections that take the largest percentage of the total time are the candidates for optimization.
- After the changes are made, re-run the business rule and compare the log messages.

> **Note**
>
> Log messages are limited to 20,000. Anything over 20,000 messages is ignored.

Example Business Rule

- In this example, we added five year-to-date members to the *Account* dimension: "4110_UTD", "4120_YTD", "4130_YTD", "4140_YTD", "4150_YTD". We then created a business rule:

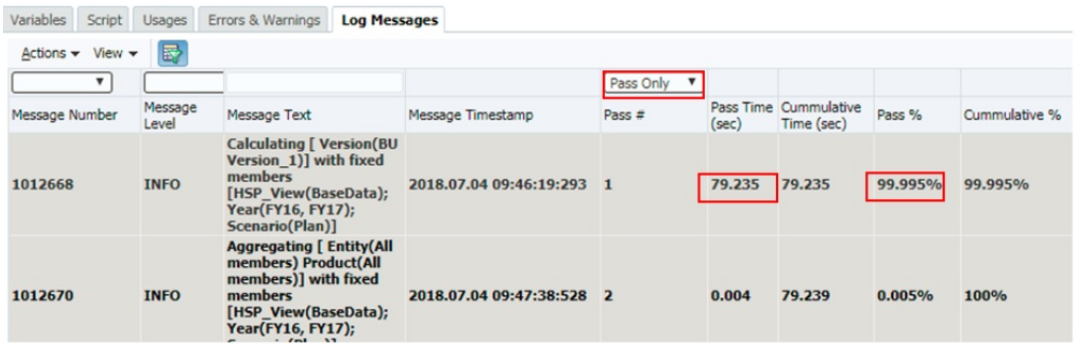
```
SET UPDATECALC OFF;
FIX ("BaseData","Plan","FY17", "FY16")
"BU Version_1"(
IF(@ismbr("Jan"))
"4110_YTD" = "4110"; "4120_YTD" = "4120"; "4130_YTD" = "4130"; "4140_YTD" = "4140"; "4150_YTD" = "4150";
Else
"4110_YTD"="4110" + @prior("4110_YTD"); "4120_YTD"="4120" + @prior("4120_YTD");
"4130_YTD"="4130" + @prior("4130_YTD"); "4140_YTD"="4140" + @prior("4140_YTD");
"4150_YTD"="4150" + @prior("4150_YTD");
Endif)
Agg("Entity","Product");
ENDFIX
```

Pass 1

Pass 2

This business rule calculates two YTD accounts and then aggregates the values through the *Product* and *Entity* dimensions.

- Looking at the log messages with *Pass Only* selected, we can see that most of the rule time is taken with the first pass.



Message Number	Message Level	Message Text	Message Timestamp	Pass #	Pass Time (sec)	Cummulative Time (sec)	Pass %	Cummulative %
1012668	INFO	Calculating [Version(BU Version_1)] with fixed members [HSP_View(BaseData); Year(FY16, FY17); Scenario(Plan)]	2018.07.04 09:46:19:293	1	79.235	79.235	99.995%	99.995%
1012670	INFO	Aggregating [Entity(All members) Product(All members)] with fixed members [HSP_View(BaseData); Year(FY16, FY17);	2018.07.04 09:47:38:528	2	0.004	79.239	0.005%	100%

- Deselecting *Pass Only* allows us to see the number of block reads/writes in the pass1 section of the logs.

Message Text	Message Timestamp	Pass #
Calculator Information Message: Total Block Created: [3.1900e+03] Blocks Sparse Calculations: [3.2000e+03] Writes and [3.2100e+03] Reads Dense Calculations: [0.0000e+00] Writes and [0.0000e+00] Reads Sparse Calculations: [0.0000e+00] Cells Dense Calculations: [0.0000e+00] Cells	2018.06.29 10:47:34:...	1

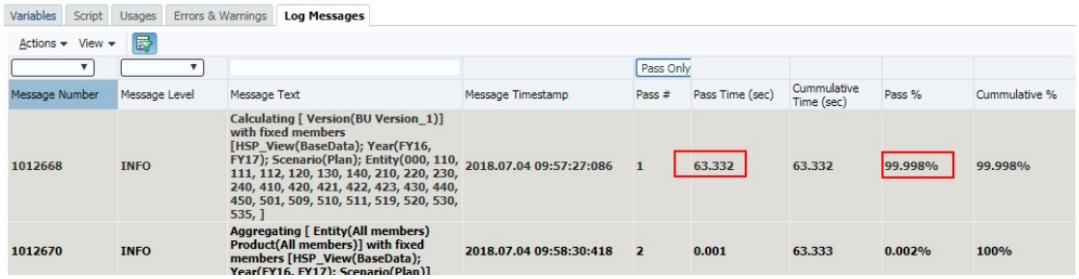
- Reviewing the syntax, we can see that the calculation does not have a fix on *Entity* and *Product*; therefore, all of the rule is being performed on all the levels of *Entity* and *Product*. There is not requirement to calculate upper levels in pass1 because these will be overwritten by the AGG statement in the second pass anyway.

```
SET UPDATECALC OFF;
FIX ("BaseData","Plan","FY17“, “FY16”)
Pass 1
"BU Version_1"(
IF(@ismbr("Jan"))
"4110_YTD" = "4110"; "4120_YTD" = "4120"; "4130_YTD" = "4130"; "4140_YTD" = "4140"; "4150_YTD" = "4150";
Else
"4110_YTD"="4110" + @prior("4110_YTD"); "4120_YTD"="4120" + @prior("4120_YTD");
"4130_YTD"="4130" + @prior("4130_YTD"); "4140_YTD"="4140" + @prior("4140_YTD");
"4150_YTD"="4150" + @prior("4150_YTD");
Endif)
Pass 2
Agg("Entity","Product");
ENDFIX
```

- The rule is reviewed and an extra Fix statement is added to add lev0 of *Entity* and *Product* to Pass1.

```
SET UPDATECALC OFF;
FIX ("BaseData","Plan","FY17")
Pass 1
Fix(@LEVMBRS("Entity",0), @LEVMBRS("Product",0))
"BU Version_1"(
IF(@ismbr("Jan"))
"4110_YTD" = "4110";
"4120_YTD" = "4120";
Else
"4110_YTD"="4110" + @prior("4110_YTD");
"4120_YTD"="4120" + @prior("4120_YTD");
Endif)
ENDFIX
Pass 2
Agg("Entity","Product");
ENDFIX
```

- The rule is then run again.
- With the change, Pass1 still takes the longest time, but now is 20% faster because upper level blocks are no longer calculated unnecessarily. The overall calculation time also improves by 20%.



Variables | Script | Usages | Errors & Warnings | **Log Messages**

Actions ▾ View ▾ | Pass Only

Message Number	Message Level	Message Text	Message Timestamp	Pass #	Pass Time (sec)	Cummulative Time (sec)	Pass %	Cummulative %
1012668	INFO	Calculating [Version(BU Version_1)] with fixed members [HSP_View(BaseData); Year(FY16, FY17); Scenario(Plan); Entity(000, 110, 111, 112, 120, 130, 140, 210, 220, 230, 240, 410, 420, 421, 422, 423, 430, 440, 450, 501, 509, 510, 511, 519, 520, 530, 535,]	2018.07.04 09:57:27:086	1	63.332	63.332	99.998%	99.998%
1012670	INFO	Aggregating [Entity(All members) Product(All members)] with fixed members [HSP_View(BaseData); Year(FY16, FY17); Scenario(Plan)]	2018.07.04 09:58:30:418	2	0.001	63.333	0.002%	100%

- Deselecting *Pass Only*, we can see that the number of blocks, reads, and writes in the Pass1 section of the logs goes down after the change.

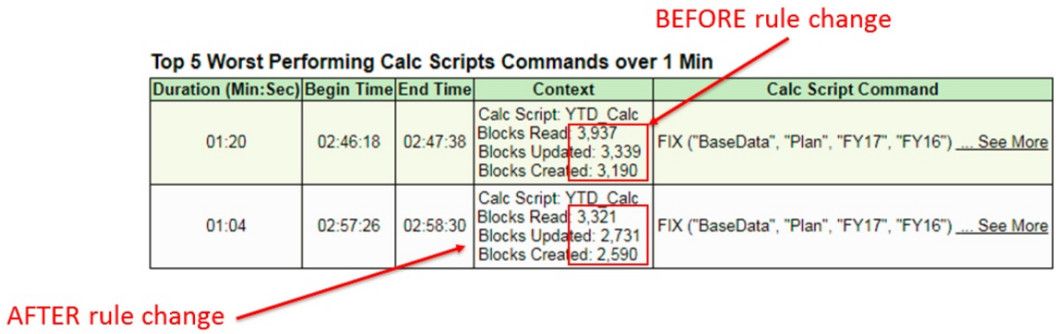
BEFORE rule change

Message Text	Message Timestamp	Pass #
Calculator Information Message: Total Block Created: [3.1900e+03] Blocks Sparse Calculations: [3.2000e+03] Writes and [3.2100e+03] Reads Dense Calculations: [0.0000e+00] Writes and [0.0000e+00] Reads Sparse Calculations: [0.0000e+00] Cells Dense Calculations: [0.0000e+00] Cells	2018.06.29 10:47:34:...	1

AFTER rule change

Message Text	Message Timestamp	Pass #
Calculator Information Message: Total Block Created: [2.5900e+03] Blocks Sparse Calculations: [2.5920e+03] Writes and [2.5940e+03] Reads Dense Calculations: [0.0000e+00] Writes and [0.0000e+00] Reads Sparse Calculations: [0.0000e+00] Cells Dense Calculations: [0.0000e+00] Cells	2018.06.29 10:38:22:...	1

- This reduces the reads and writes for the overall business rule (both passes), as we can see in the Activity Report.



4

Designing Business Rule Sets

Related Topics

- About Business Rulesets
  You create a business ruleset by combining business rules (or business rulesets) that can be launched simultaneously or sequentially.
- Creating a Business Ruleset
  You can create a business ruleset from the System, Filter, Custom, and Deployment views, and from the Ruleset and Template Designers.
- Opening a Business Ruleset
  You can open a business ruleset from the System, Filter, Deployment, and Custom View.
- Opening a Business Rule Within a Business Ruleset
  You can open a business rule from within a business ruleset.
- Adding a Business Rule to a Business Ruleset
  When you add a business rule to a business ruleset, the rules in the ruleset can be launched sequentially or simultaneously.
- Removing a Business Rule from a Business Ruleset
  When you remove a business rule from a business ruleset, the rule is not deleted. The rule exists independently of the ruleset in the database.
- Copying a Business Ruleset to Another Application
  When you copy a business ruleset to another application, ensure that you copy it to the same application type.
- Savings Business Rulesets
  When you save a business ruleset, it is saved to the application and application type for which you created it. .
- Deleting a Business Ruleset
  You can delete a business ruleset only if it is not being used by other business rulesets.

About Business Rulesets

You create a business ruleset by combining business rules (or business rulesets) that can be launched simultaneously or sequentially.

Your ability to create rulesets is determined by the role you are assigned. See *Using Oracle Planning and Budgeting Cloud*.

After you create and save a ruleset, you validate and deploy it in Calculation Manager. Then you can launch it from your application.

> **Tip**
>
> As you create business rulesets, you may want to leave the business rules, components, templates and variables you're working with open. Calculation Manager displays these objects in a tabbed interface so you can move easily among the tabs as you are creating business rulesets. You can have as many as ten tabs open within Calculation Manager, but Oracle recommends that you not open more than ten objects simultaneously for optimum performance.

Creating a Business Ruleset

You can create a business ruleset from the System, Filter, Custom, and Deployment views, and from the Ruleset and Template Designers.

To create a business ruleset:

1. Right-click **RuleSets**, and then select **New**.
2. In the **New RuleSet** dialog box, enter a name and select and application type and application, and then click **OK**.

   If you are adding a ruleset from the System View, the application type and application are automatically populated.
3. From **Existing Objects**, drag existing rules and rulesets and drop them into the flow chart.

   Planning rulesets can contain rules and rulesets that are created in and deployed to different applications.
4. Do any of the following tasks:
   - To change the order of the rules in the ruleset, click the up or down arrows, or right-click the rule, and then select **Move Up** or **Move Down**.
   - To remove a rule from the ruleset, right-click the rule, and then select **Remove**.
   - To open a business rule for editing, right-click the rule, and then select **Open**.
5. In **Properties**, enter properties for the ruleset.

   In the RuleSet Designer, if you select a rule within the ruleset you are creating, its properties are displayed in **Properties** instead of the ruleset's properties.

   Select **Enable Parallel Execution** if you want the rules and rulesets in the ruleset to launch simultaneously. By default the rules and rulesets belonging to a ruleset launch sequentially - each rule or ruleset in the ruleset must run without errors before the next rule or ruleset is launched.

   If the ruleset contains nested rulesets, and the nested rulesets have a different **Enable Parallel Execution** setting than the parent ruleset, the setting of the nested ruleset applies. For example, if you have ruleset1 (that is flagged for *parallel* processing) and it contains rule1, rule2, and ruleset2 (flagged for *sequential* processing), the rules and rulesets in ruleset2 are processed sequentially, even though ruleset1 is flagged for parallel processing.
6. On the **Usages** tab, review what rulesets are using this ruleset, if any.

   By default, a ruleset is used by no other rulesets when you create it so this tab is empty when you initially create the business ruleset.

   If you select a rule in the ruleset, you can see the names of the business rulesets.

7. On the **Variables** tab, review and enter information about the variables used in the ruleset.

   Select **Merge Variables** to merge all instances of the *same variable* used in the rules within this ruleset so only the first instance of each variable is displayed when the rule is launched. If you do not merge variables, all instances of each variable are displayed.

   If you select **Merge Variables**, the first value that the user enters for the runtime prompt is used for all subsequent occurrences of that runtime prompt during validation and launch.

8. Click  .

Opening a Business Ruleset

You can open a business ruleset from the System, Filter, Deployment, and Custom View.

To open a business ruleset, do one of the following:

- Right-click the ruleset, and then select **Open**.
- Double-click the ruleset.

Opening a Business Rule Within a Business Ruleset

You can open a business rule from within a business ruleset.

- In **System View**, expand **RuleSets**, then expand the desired ruleset, and then double-click the desired rule.
- With a business ruleset open for editing, double-click the desired rule within the ruleset.

Adding a Business Rule to a Business Ruleset

When you add a business rule to a business ruleset, the rules in the ruleset can be launched sequentially or simultaneously.

To add a business rule to a business ruleset:

1. Open a ruleset.
2. From **Existing Objects**, drag existing rules into the ruleset.
3. Click  .

Removing a Business Rule from a Business Ruleset

When you remove a business rule from a business ruleset, the rule is not deleted. The rule exists independently of the ruleset in the database.

To remove a business rule from a business ruleset:

1. Open the ruleset.
2. Right-click the desired rule, and then select **Remove**.

Copying a Business Ruleset to Another Application

When you copy a business ruleset to another application, ensure that you copy it to the same application type.

To copy a business ruleset to another application:

1. In **System View**, right-click a business ruleset, and then select **Copy To**.
2. In **Copy To**, enter a name for the business ruleset, or accept the default name, and select an application type and application..

   You cannot copy a business ruleset to more than one application type and application.
3. Select whether to copy any shared components and whether to replace existing components with the same names.

   These options are enabled only if there are shared components in the business ruleset you want to copy to the new application.
4. Click **OK**.

   The new business ruleset is added to the application and application type you selected. To see it in the System View, you may need to refresh the application list. To refresh the application list, right-click the application, and then select **Refresh**. See Refreshing Business Rules or Business Rulesets.

Savings Business Rulesets

When you save a business ruleset, it is saved to the application and application type for which you created it. .

After you save a ruleset, you can deploy, validate, and launch it

You can deploy and validate a business ruleset in Calculation Manager, and you can launch the ruleset from Planning.

To save a business ruleset after you create or edit it, click  .

> **Note**
>
> To see the business ruleset in the **System View** after you save it, you may need to refresh the application list. To do this, right-click any node or object in the System View, and then select **Refresh**.

Deleting a Business Ruleset

You can delete a business ruleset only if it is not being used by other business rulesets.

If the ruleset is being used, you must remove it from the business rulesets that are using it or make copies of it for the business rulesets that are using it before you delete it.

To delete a business ruleset:

1. In **System View**, right-click the ruleset, and then select **Show Usages** to ensure that it is not being used by other rulesets.
2. If the ruleset is not being used by other rulesets, right-click the ruleset, and then select **Delete**.
3. Click **OK** to confirm deletion of the ruleset.

5

Working with System Templates

Related Topics

- About System Templates
  System templates are predefined templates that perform calculations.
- Displaying System Templates
  Where system templates are displayed depends on whether they are *graphical* templates or *rule* templates.
- Using System Templates
  To use a system template, selected the template and drag it into a flow chart or script.
- Showing the Template Flow
  When you are reviewing the script generated by a system template, it is sometimes helpful to see the template flow.
- Saving a System Template as a Custom Template
  You may want to customize the content of a system template.
- Removing a System Template from a Business Rule
  You can remove system templates from business rules.

About System Templates

System templates are predefined templates that perform calculations.

Calculations preformed by system templates include clearing data; copying data; calculating amounts, units and rates; distributing data; allocating values; aggregating data; entering script commands to optimize performance; and converting account values to reporting currencies.

You can include system templates in:

- Graphical or script business rules
- Graphical or script custom templates

As a component of a business rule or custom template, a system template contains a series of steps into which you enter parameters. These parameters, combined with the template logic, generate a calculation script within the business rule or template. This makes the templates easier to use, and reduces potential calculation script syntax errors.

In Calculation Manager, all system templates are available as wizards. Working with system templates in a wizard makes the templates easier to use, and reduces potential calculation script syntax errors.

The prompts in system templates are filtered based on the choices made in the wizard. For example, the Aggregation system template displays a step for selecting dense dimensions to aggregate, but if there are no dense dimensions available for aggregation, then the steps for full and partial dense dimensions are not displayed.

> **Note**
>
> You can use system templates in Planning block storage applications.

Displaying System Templates

Where system templates are displayed depends on whether they are *graphical* templates or *rule* templates.

When you create or open a *graphical* rule or template, system templates are displayed in either:

- **New Objects**—System templates are displayed in **New Objects** as individual objects.
- **Existing Objects**—System templates are displayed in **Existing Objects** under the **System Templates** header.

When you create or open a *script* rule or template, system templates are displayed only in **Existing Objects**.

To access the wizard for a system template:

- For a *graphical* rule or template, select the desired template and drag it into the flow chart between **Begin** and **End**.
- For a *script* rule or template, select the template and drag it into the script.

Using System Templates

To use a system template, selected the template and drag it into a flow chart or script.

Related Topics

- Using the Clear Data Template
  Use the Clear Data template to clear data from members in the cube.
- Using the Copy Data Template
  Use the Copy Data template to copy data from one location in the database to another.
- Using the Amount-Unit-Rate Template
  Use the Amount-Unit-Rate template to calculate one of three members that you input for amounts, units and rates.
- Using the Allocate - Level to Level Template
  Use the Allocate Level to Level template to allocate from one level to another in the database outline.
- Using the Allocation Template
  Use the Allocation template to allocate values from a source to a destination, either evenly or based on a specified driver.
- Using the Aggregation Template
  Use the Aggregation template to aggregate data values of members you specify.
- Using the SET Commands Template
  Use the SET Commands template to enter script commands that optimize the performance of calculation scripts.

- Using the Currency Conversion Template
  The Currency Conversion template converts account values to reporting currencies using system type accounts or accounts that you select.

Using the Clear Data Template

Use the Clear Data template to clear data from members in the cube.

To clear data, you specify the members whose values you want to clear. You can clear data for one member or for a block of members.

For example, you could use the Clear Data template to clear forecast data before copying data from "actual" to "forecast" and making changes.

To use the Clear Data template:

1. Create or open a business rule or a template.
2. Do one of the following:
   - For a *graphical* rule or template, under **New Objects**, select **Clear Data**, and then drop it into the flow chart between **Begin** and **End**.
   - For a *script* rule or template, expand **System Templates**, then select **Clear Data**, and then drop it into the desired area in the script.
3. Enter the requested information in the **Clear Data Wizard**.

Table 5-1 Clear Data Wizard Steps

Step	Explanation
Data Selection	Define the data to clear by selecting one or more members for each dimension. The following options are available: • Click  next to **Use Predefined Selection** to select variables to populate all the dimensions with values. • Click  next to **Member Selector** to select members and functions for each dimension. • Select a dimension, and then click  to select a variable, member, or function. In **Select a Clearblock Option**, select an option to define how to clear the data: • **All**—Clears all blocks of data • **Upper**—Clears only consolidated blocks of data. • **Dynamic**—Clears blocks containing values derived from Dynamic Calc and Store member combinations • **Non-Input**—Clears data blocks containing derived values. This works only for blocks that are created completely by a calculation operation and not blocks into which any values are loaded. • **N/A**—Clears data blocks and sets the members to #Missing
Settings	If you have a multi-currency application with `HSP_Rates` set to "sparse" or a single-currency application, **Settings** displays the members selected in **Data Selection**. Click the drop-down, and select a dimension where a single member is selected, preferably a dense dimension. **Note:** Since the script uses the member from the dimension you select, if you cannot select a dimension where a single member is selected, then you cannot use this template.

Using the Copy Data Template

Use the Copy Data template to copy data from one location in the database to another.

For example, you could use the Copy Data template to create a budget for 2015 by copying the values for your accounts and entities from 2014. In this case, you would copy the 2014 actuals to the forecast for 2015. You could also use the Copy Data template to copy budget data from a "worst case " budget scenario to a "best case" scenario, where you can make changes.

To use the Copy Data template :

1. Create or open a business rule or a template.
2. Do one of the following:
   - For a *graphical* rule or template, under **New Objects**, select **Copy Data**, and then drop it into the flow chart between **Begin** and **End**.
   - For a *script* rule or template, expand **System Templates**, then select **Copy Data**, and then drop it into the desired area in the script.
3. Enter the requested information in the **Copy Data Wizard**.

Table 5-2 Copy Data Wizard Steps

Steps	Explanation
Information	Describes the function of the template.
Point of View	Define the data to copy by selecting one or more members for each dimension. Leave the dimensions that vary for copy from and to empty. You specify these members in the following steps The following options are available: • Click  next to **Use Predefined Selection**, to select variables to populate all the dimensions with values. • Click  next to **Member Selector**, to select members and functions for each dimension. • Select a dimension, and then click  to select a variable, member, or function.
Copy From	Select the members from which to copy data. **Note:** Select one member for each dimension. If you do not select a member for each dimension, a warning message is displayed.
Copy To	Select the member to which to copy data.
Options	**Optional:** Enter a percentage increase or decrease to apply to the destination range of data cells. For example, to increase the target data by 10 percent, enter **10**, and to decrease the target data by 10 percent, enter **-10**. To enter a variable, click  . For example, you could define a run-time prompt variable, and then enter the value for this increase or decrease at runtime.

Using the Amount-Unit-Rate Template

Use the Amount-Unit-Rate template to calculate one of three members that you input for amounts, units and rates.

For example, if you want to calculate product revenue and you know the quantity and price, you select **Calculate Amounts**, and enter the product revenue for amounts, the quantity for units, and the price for rates. This calculates product revenue by multiplying quantity and price.

To use the Amount-Unit-Rate template:

1. Create or open a business rule or a template.
2. Do one of the following:
   - For a *graphical* rule or template, under **New Objects**, select **Amount-Unit-Rate**, and then drop it into the flow chart between **Begin** and **End**.
   - For a *script* rule or template, expand **System Templates**, then select **Amount-Unit-Rate**, and then drop it into the desired area in the script.
3. Enter the requested information in the **Amount-Unit-Rate Wizard**.

Table 5-3 Amount-Unit-Rate Wizard Steps

Steps	Explanation
Information	Describes the function of the template.
Select Calculation Method	Define a calculation method to use by selecting one of the following options from the drop-down list: • **Calculate Amounts**—`Amounts = Units * Rates` • **Calculate Units**—`Units = Amounts / Rates` • **Calculate Rates**—`Rates = Amounts / Units` • **Calculate All**—Select if the member with `#Missing` may vary or is unknown. This calculation method searches for the `#Missing` values and applies the appropriate calculation method to solve for these unknown values.
Point of View	Define the members that do not vary in the allocation process. The following options are available for selecting members. • Click  next to **Use Predefined Selection**, to select variables to populate all the dimensions with values. • Click  next to **Member Selector**, to select members for each dimension. • Select a dimension, and then click  to select a variable, member, or function.

Table 5-3 (Cont.) Amount-Unit-Rate Wizard Steps

Steps	Explanation
Calculate Amounts/ Calculate Units / Calculate Rates / Calculate All	This step varies depending on what you selected for the calculation method. • If you select **Calculate Amounts**, enter a member or click to select to select a member for Amounts, then click Next to enter members for Units, and then click Next to enter members for Rates. On the Units and Rates steps, you can enter members that vary from the Point of View. • If you selected to **Calculate Units**, enter a member or click to select a member for Units, then click Next to enter members for Amounts, and then click Next to enter members for Rates. On the Amounts and Rates steps, you can enter members that vary from the Point of View. • If you selected to **Calculate Rates**, enter a member or click to select a member for Rates, then click Next to enter members for Amounts, and then click Next to enter members for Units. On the Amounts and Units steps, you can enter members that vary from the Point of View • If you select **Calculate All**, enter or select members for Amounts, Units, and Rates.

Using the Allocate - Level to Level Template

Use the Allocate Level to Level template to allocate from one level to another in the database outline.

For example, you could use the Allocate - Level to Level template to allocate products from one level to another. Assume you have a the following product outline:

```
T_TP
    P_000
    T_TP1
   P_100
           P_110
   etc
     T_TP2
          P_200
          P_210
   etc
     T_TP3
   P_291
           P_292
```

In this example, you would use level/generation 3 for the start and level/generation 4 for the end in the Allocate - Level to Level template to allocate from the parent members (`T_TP1`, `T_TP2`, `T_TP3`) to their children (level 0) members.

To use the Allocate - Level to Level template:

1. Create or open a business rule or a template.
2. Do one of the following:
   - For a *graphical* rule or template, under **New Objects**, select **Allocate - Level to Level**, and then drop it into the flow chart between **Begin** and **End**.

- For a *script* rule or template, expand **System Templates**, then select **Allocate - Level to Level**, and then drop it into the desired area in the script.

3. Enter the requested information in the **Allocate - Level to Level Wizard**.

Table 5-4 Allocate - Level to Level Wizard Steps

Steps	Explanation
Point of View	Select one or more members for each dimension listed that you do not want to vary during the allocation. The following options are available: • Click  next to **Use Predefined Selection**, to select variables to populate all the dimensions with values. • Click  next to **Member Selector**, to select members and functions for each dimension. • Select a dimension, and then click  to select a variable, member, or function.
Source	For each dimension listed, select the member whose data you want to allocate. After you select the members, enter the percentage of the source to allocate. **Note:** Leave the dimension to use for the level to level allocating empty. (You select this dimension in the next step.)
Allocation Range	Define the allocation range by entering the requested information. The rule allocates all data below the selected member, from the level specified as start level, down to the level specified as end level.
Target	• If dimensions have been defined on the **Source** step and have not been used for allocation, define a target member for each dimension. The members you selected on the **Source** are entered here by default. Use one of these options to define target members: – Click  next to **Use Predefined Selection**, to select variables to populate all the dimensions with values. – Click  next to **Member Selector**, to select members and functions for each dimension. – Select a dimension, and then click  to select a variable, member, or function. For optimization, select a dimension that has only one member. • Select the dimension from which you selected a single member above.
Offset	Define the offset or leave the table in this step blank if you don't want to generate an offset calculation. If you do define an offset, do the following: • Define the offset's dimensionality To write the *total* amount of allocated values to an offset member, select a member to define the offset. • Define the offset member on the allocation dimension Type in a member, or click  to select a member.

Table 5-4 (Cont.) Allocate - Level to Level Wizard Steps

Steps	Explanation
Driver	Define the driver (basis) used to calculate the percentage applied to each member of the allocation range. • Select members only for dimensions that vary from the Point of View. Use one of these options to select members: – Click  next to **Use Predefined Selection**, to select variables to populate the dimensions with values. Select only one member per dimension. – Click  next to **Member Selector**, to select members and functions for each dimension. – Select a dimension, and then click  to select a variable, member, or function. • Specify whether to update the driver's aggregations on the allocation dimension.
Other Options	Answer the questions in the wizard.

Using the Allocation Template

Use the Allocation template to allocate values from a source to a destination, either evenly or based on a specified driver.

For example, you could use the Allocation template to allocate administrative expenses to the level 0 members of the Product dimension, using Gross Sales as the basis.

To use the Allocation template:

1. Create or open a business rule or a template.
2. Do one of the following:
   - For a *graphical* rule or template, under **New Objects**, select **Allocation**, and then drop it into the flow chart between **Begin** and **End**.
   - For a *script* rule or template, expand **System Templates**, then select **Allocation**, and then drop it into the desired area in the script.
3. Enter the requested information in the **Allocation Wizard**.

Table 5-5 Allocation Wizard Steps

Step	Explanation
Information	Describes the function of the template.

Table 5-5 (Cont.) Allocation Wizard Steps

Step	Explanation
Point of View	Define the members that do not vary in the allocation. You must select at least one member from one dimension. The following options are available to select members: • Click  next to **Use Predefined Selection**, to select variables to populate all the dimensions with values. • Click  next to **Member Selector**, to select members for each dimension. • Select a dimension, and then click  to select a variable, member, or function.
Source	For each dimension, select a member from which to allocate the data, and enter the percentage of the source to allocate. For example, to allocate 25% of the source value enter 25.
Allocation Range	Select the dimension to which to allocate the data, and the parent member for this dimension. The data is allocated to the level 0 members below the specified parent member in the database outline. (The level 0 member is the lowest member in the outline with no members beneath it.)
Destination - Target	Specify a target member for each dimension defined in the source.
Destination - Offset	**Optional.** • Specify the offset to generate an offsetting calculation by entering a member for each dimension. An offsetting value can be calculated as either the same or opposite value of the source. For example, if you have $1000 in your source to allocate, you can use an offset so that you don't double your numbers. When you run the rule, $1000 will be allocated to level 0, and the offset will create the -1000 in the members specified. • Specify whether to clear the offset data before the allocation process begins. • Specify whether the offset's value should be the opposite sign of the allocated data.
Exclude	**Optional.** Select members to exclude from the allocation range. These members do not receive any allocated data, and their driver values are excluded from the driver percentage basis.
Driver	Define whether to "allocate using a driver" or "allocate evenly". Select one member per dimension for each dimension that varies from the Point of View. Dimensions you leave empty are inherited from the Point of View and from the upper member ranges defined in the rule.
Spread Method	If you selected "allocate evenly" in the **Driver** step, select an option to spread the data.

Table 5-5 (Cont.) Allocation Wizard Steps

Step	Explanation
Other Options	Answer the questions in the wizard. If you select a rounding method, there are additional steps depending on the rounding method you select. The following rounding methods are available: • **Define Rounding Member**—If you select this option, then next step is **Rounding on Member**, where you enter the number of decimal places to use for the allocation and select a member on which to place the allocation rounding difference. • **Use Biggest Value**—If you select this option, the next step is **Round on Biggest**, where you enter the number of decimal places to use for the allocation. • **Use Smallest Value**—If you select this option, the next step is **Round on Smallest**, where you enter the number of decimal places to use for the allocation. • **No Rounding**—If you select this option, there are no additional steps.

Tutorial

Your Goal	Complete This Tutorial
Learn how to allocate Planning and Budgeting costs using the Allocation System Templates in Calculation Manager. This is a no-script-needed approach to allocating costs.	Allocating Costs Using the Allocation Template in Calculation Manager

Using the Aggregation Template

Use the Aggregation template to aggregate data values of members you specify.

Using the Aggregation template, you can:

- Restrict the aggregation by entering members in the Point of View
- Specify whether to aggregate missing values
- Specify whether to aggregate values into the local currency (not supported for hybrid aggregation)

When aggregating values, the following calculation commands are supported:

- SET UPTOLOCAL ON: Aggregate the data up to the local currency (This commnad is not suppored for hybrid aggregation.)
- SET CACHE HIGH: Set a value for the calculator cache (This command is not supported for hybrid aggregation.)
- SET AGGMISSG ON: Aggregate the missing values in the database
- SET FRMLBOTTOMUP ON: Optimize the calculation on a sparse dimension

For example, you could use the Aggregation template to aggregate dense or sparse dimensions using a parent member, and select a level of aggregation, such as Descendants, Children, or Ancestors(all).

To use the Aggregation template:

1. Create or open a business rule or a template.

2. Do one of the following:
   - For a *graphical* rule or template, under **New Objects**, select **Aggregation**, and then drop it into the flow chart between **Begin** and **End**.
   - For a *script* rule or template, expand **System Templates**, then select **Aggregation**, and then drop it into the desired area in the script.
3. Enter the requested information in the **Aggregation Wizard**.

Table 5-6 Aggregation Wizard Steps

Step	Explanation
Information	Describes the function of the template.
Point of View	Select the members for the dimensions to which you want to restrict the aggregation. The following options are available to select members. • Click  next to **Use Predefined Selection**, to select variables to populate all the dimensions with values. • Click  next to **Member Selector**, to select members for each dimension. • Select a dimension, and then click  to select a variable, member, or function.
Full Dense Aggregation	Select up to two dense dimensions to aggregate fully, and specify whether the dense dimensions have stored non-level 0 members. Note the following: • If a dense dimension has stored non-level 0 members, then a Calc Dim (dense dimension) is created in the script. • If a dense dimension does not have stored non-level 0 member, then no script is generated for the dense dimension. • If you want to do a partial aggregation on a dense dimension, (parent member with function - Descendants, Children, or Ancestors), then do not select the dimensions in this step.
Full Sparse Aggregation	Select up to three sparse dimensions to aggregate fully, and specify whether the sparse dimensions have member formulas that need to be calculated. If a sparse dimension has a member formula, then the sparse dimension is calculated. If a sparse dimension does not have a member formula, then the sparse dimension is aggregated.
Partial Dimension Aggregation Dense	Select up to two dense dimensions for partial aggregation. For each dense dimension, select a member and an aggregation level. Note the following: • The Partial Dimension Aggregation Dense step is displayed if you have a dense dimension that you did not use on the Full Dense Aggregation step. If you use two dense dimensions on the Full Dense Aggregation tab, and there are no more dense dimensions in the application, then the Partial Dimension Aggregation Dense step is not displayed. • If you want to do a partial aggregation on a sparse dimension, (parent member with function - Descendants, Children, or Ancestors), then do not select the dimension on this step.

Table 5-6 (Cont.) Aggregation Wizard Steps

Step	Explanation
Partial Dimension Aggregation Sparse	Select up to three sparse dimensions for partial aggregation. For each sparse dimension, select a member and an aggregation level. Note the following: **Note:** • The available sparse dimension for partial aggregation are displayed in gray. • The Partial Dimension Aggregation Sparse step is displayed if you have a sparse dimension that you did not use on the Full Sparse Aggregation step. If you use three sparse dimensions on the Full Sparse Aggregation tab, and there are no more sparse dimensions in the application, then the **Partial Dimension Aggregation Sparse** step is not displayed. • If you want to do a partial aggregation on a dense dimension, (parent member with function - Descendants, Children, or Ancestors), then do not select the dimension on this step.
Settings	Specify the settings on the step.

Using the SET Commands Template

Use the SET Commands template to enter script commands that optimize the performance of calculation scripts.

You can include data volume, data handling, memory usage, and threading and logging script commands.

To use the SET Commands template:

1. Create or open a business rule or a template.
2. Do one of the following:
   - For a *graphical* rule or template, under **New Objects**, select **SET Commands** and then drop it into the flow chart between **Begin** and **End**.
   - For a *script* rule or template, expand **System Templates**, then select **SET Commands**, and then drop it into the desired area in the script.
3. Enter the requested information in the **Set Commands Wizard**.

   The wizard includes the following steps:

   - **Data Volume**
   - **Data Handling**
   - **Memory Usage**
   - **Threading**
   - **Logging**

   Each step has several questions that can be answered by selecting an option in the drop-down list next to the question. you can select one or more options in this template.

> **Note**
>
> You do not need to answer every question in the Set Commands wizard. You can answer only the questions that are applicable to your situation.

> **Note**
>
> When the application is in Essbase Hybrid, only the **Data Volume**, **Data Handling**, and **Threading** tabs are displayed in the wizard.

Using the Currency Conversion Template

The Currency Conversion template converts account values to reporting currencies using system type accounts or accounts that you select.

You use the Currency Conversion template in Planning applications that are created with multiple currencies selected.

For an example of using the Currency Conversion template, see Currency Conversion Template Example.

> **Note**
>
> If the Planning application is created with multi-currencies, the Currency Conversion template does not display in the list of System Templates.

To use the Currency Conversion template:

1. Create or open a business rule or a template.
2. Do one of the following:
   - For a *graphical* rule or template, under **New Objects**, select **Currency Conversion**, and then drop it into the flow chart between **Begin** and **End**.
   - For a *script* rule or template, expand **System Templates**, then select **Currency Conversion**, and then drop it into the desired area in the script.
3. Enter the requested information in the **Currency Conversion Wizard**.

Table 5-7 Currency Conversion Wizard Steps

Step	Explanation
Information - Currency	Describes the function of the template and what information you need to use the template.

Table 5-7 (Cont.) Currency Conversion Wizard Steps

Step	Explanation
Exchange Rate Option	Answer the questions in the wizard about currency: • **In what dimension is your currency?** Select the dimension that corresponds to your currency dimensions. • **Select the reporting currency.** Select the reporting currency to which you want to convert. • **Select the parent member that contains your currency members.** Select the parent member that is the parent for all of the currencies used in your application; for example, "Input Currencies. " • **What exchange rate (currency) is used for this reporting currency?** Select the currency that corresponds to the reporting currency you selected above. For example, if you selected *EUR Reporting* for the reporting currency, you would select *EUR* (where *EUR* is the currency used to convert data to *EUR Reporting*). • **Do you want to use the account types for currency conversion?** Select **Yes** to use account types for currency conversion. If you select **Yes**, Expense and Revenue accounts will use average exchange rates and Liability and Equity accounts will using ending exchange rates for conversion. If you select **No**, you are prompted later in the wizard to specify what accounts to use for the average and ending exchange rates • **Do you want to calculate average exchange rates?** If you select **Do Not Calculate Average**, the steps related to average are not displayed, and the script for calculating currency based on average exchange rates is not generated. • **Do you want to calculate ending exchange rates?** Select whether to calculate ending exchange rates. If you select **Do Not Calculate Ending**, the steps related to ending are not displayed, and the script for calculating currency based on ending exchange rates is not generated.
POV	Select the members to use in the conversion to the reporting currency. The following options are available to select members. • Click  next to **Use Predefined Selection**, to select variables to populate all the dimensions with values. • Click  next to **Member Selector**, to select members for each dimension. • Select a dimension, and then click  to select a variable, member, or function.
Accounts Average	Enter the account(s) to use the average exchange rate to convert to the reporting currency. **Note:** This step is displayed only if you selected **No** in **Do you want to use the account types for currency conversion?** on the **Exchange Rate Option** step.
Accounts Ending	Enter the account(s) to use the ending exchange rate to convert to the reporting currency. **Note:** This step is displayed only if you selected **No** in **Do you want to use the account types for currency conversion?** on the **Exchange Rate Option** step.
FX Average	Select the members that contain the average exchange rates. **Note:** You only need to select the members that are different than the members in the Point of View.

Table 5-7 (Cont.) Currency Conversion Wizard Steps

Step	Explanation
FX Ending	Select the members that contain the ending exchange rates. **Note:** You only need to select the members that are different than the members in the Point of View.

Currency Conversion Template Example

1. In a Planning application that was created with no multiple currencies in addition to the existing dimensions, create a custom dimension named "Currencies" and add these members:
   - Input Currencies
     - USD
     - CAD
     - EUR
     - GBP
     - INR
   - Reporting Currencies
     - EUR Reporting
     - US Reporting
     - Can Reporting
2. In the "Account" dimension, add these members for FX_Rates:
   - FX_Average
   - FX_Ending
3. In the "Entity" dimension, add the entity "Company Assumptions".
4. Using Oracle Smart View for Office or a Planning form, enter exchange rates in these cells:
   - **Account**—FX_Average and FX_Ending
   - **Period**—Enter rates for each month
   - **Year**—FY12
   - **Scenario—**Current
   - **Version**—BU Version_1
   - **Entity**—Company Assumptions
   - **Products**—000
   - **Currencies**—Enter an exchange rate for each currency
5. In Variables:
   a. Create an application level runtime prompt using a member variable named: "Reporting_Currency" for the Currencies dimension, and enter a default value of "EUR Reporting."
   b. Create an application level runtime prompt using a member variable named: "Input Currencies" for the Currencies dimension, and enter a default value of "EUR."

c. Create a business rule in this Planning application, and drag and drop the Currency Conversion system template into the business rule's flow chart.

   The Currency Conversion Wizard is displayed.

6. In the Currency Conversion Wizard, perform these steps:

   a. **Step1 – Exchange Rate Option**

      - In what dimension is your currency? Click the dropdown, and select the Currencies dimension you created above.
      - Select the reporting currency: Click  , select **Variable**, and select the Reporting_Currency variable.

        In this example, we are using a variable for the reporting currency so that the rule can be launched in Planning for each reporting currency.
      - Select the parent member that contains your currency members: Click  , and select the Input Currencies parent member.

        The reason that you need to select a parent for the currency members is so that if a currency is added in the future, then this business rule only needs to be redeployed and launched again in Planning. The script of the template will add the new currency to the script, and calculate the reporting currency correctly.
      - What exchange rate (currency) is used for this reporting currency? Click  , and select the Input Currencies variable.

        In this example, we are using a variable for the reporting currency so that the rule can be launched in Planning for each reporting currency.
      - Do you want to use the account types for currency conversion? Using the dropdown, select **Yes**.

        If you select No, you are prompted later in the steps to specify which accounts use Average/Ending.
      - Do you want to calculate Average exchange rates?

        Using the dropdown, select **Calculate Average**.
      - Do you want to calculate Ending exchange rates?

        Using the dropdown, select **Calculate Ending**.

   b. **Step 2 – POV**

      Enter the following for each dimension:

      - **Period**—Leave period empty so that it will write to all periods
      - **Year**—"FY12"
      - **Scenario**—"Current"
      - **Version**—"BU Version_1"
      - **Entity**—@Relative("South",0)
      - **Products**—@Relative("Tennis",0),@Relative("Golf",0)

   c. **Step 3 — FX_Average**

      - Select members containing the average exchange rates:
        - **Account**—"FX_Average"
        - **Period**

- **Year**
- **Scenario**
- **Version**
- **Entity**—"Company Assumptions"
- **Products**—"000"

- Leave the Period dimension empty, so that it will use the exchange rate for each period.
- Leave Year, Scenario, and Version empty, so that they will use the members from the POV input for these dimensions.

d. **Step 4 – FX_Ending**

- Select members containing the average exchange rates:
  - **Account**—"FX_Average"
  - **Period**
  - **Year**
  - **Scenario**
  - **Version**
  - **Entity**—"Company Assumptions"
  - **Products**—"000"
- Leave the Period dimension empty, so that it will use the exchange rate for each period.
- Leave Year, Scenario, and Version empty, so that they will use the members from the POV input for these dimensions.

7. Save, validate, and deploy the business rule to Planning.

   The application in this example contains the following data:

Table 5-8 Example Data Part 1

N/A	N/A	N/A	FY12	Current	BU Version_1
N/A	N/A	Tennessee	Tennessee	Florida	Florida
N/A	N/A	Jan	Jan	Jan	Jan
N/A	N/A	G400	G100	G400	G100
Gross Sales	USD	10750	13450	9500	9699
Gross Sales	CAD	10450	10000	14450	13000
Gross Sales	EUR	40000	41450	42450	65000
Gross Sales	GBP	13250	16750	172000	16300
Gross Sales	INR	750000	750000	750000	750000
Gross Sales	US Reporting	#Missing	#Missing	#Missing	#Missing
Gross Sales	CAN Reporting	#Missing	#Missing	#Missing	#Missing
Gross Sales	EUR Reporting	#Missing	#Missing	#Missing	#Missing
Salaries	USD	5000	5000	5000	5000
Salaries	CAD	4500	4500	4500	4500
Salaries	EUR	5500	5500	5500	5705
Salaries	GBP	1200	1200	1200	1200

Table 5-8 (Cont.) Example Data Part 1

N/A	N/A	N/A	FY12	Current	BU Version_1
Salaries	INR	100000	100,000	100,000	100,000
Salaries	US Reporting	#Missing	#Missing	#Missing	#Missing
Salaries	CAN Reporting	#Missing	#Missing	#Missing	#Missing
Salaries	EUR Reporting	#Missing	#Missing	#Missing	#Missing

Notice that none of the reporting currencies in this table have values.

8. Launch the rule in Planning. For the Reporting Currency variable, enter **EUR Reporting**, and for the input currency enter **EUR**. Click **Launch**.
9. For each account in Planning, verify that **Eur Reporting** now shows the total account in the Euro:

Table 5-9 Example Data Part 2

N/A	N/A	N/A	FY12	Current	BU Version_1
N/A	N/A	Tennessee	Tennessee	Florida	Florida
N/A	N/A	Jan	Jan	Jan	Jan
N/A	N/A	G400	G100	G400	G100
Gross Sales	USD	10750	13450	9500	9699
Gross Sales	CAD	10450	10000	14450	13000
Gross Sales	EUR	40000	41450	42450	65000
Gross Sales	GBP	13250	16750	172000	16300
Gross Sales	INR	750000	750000	750000	750000
Gross Sales	US Reporting	#Missing	#Missing	#Missing	#Missing
Gross Sales	CAN Reporting	#Missing	#Missing	#Missing	#Missing
Gross Sales	EUR Reporting	52161391	52168683	52170600	52190796.69
Salaries	USD	5000	5000	5000	5000
Salaries	CAD	4500	4500	4500	4500
Salaries	EUR	5500	5500	5500	5705
Salaries	GBP	1200	1200	1200	1200
Salaries	INR	100000	100000	100000	100000
Salaries	US Reporting	#Missing	#Missing	#Missing	#Missing
Salaries	CAN Reporting	#Missing	#Missing	#Missing	#Missing
Salaries	EUR Reporting	6963318	6963318	6963318	6965471.887

10. In Planning, you can launch the business rule as many times as you need to for each reporting currency. You can change the variable inputs each time and enter the desired Reporting Currency and its corresponding exchange rate name for the variables. and for each reporting currency available in the application.

    In this example, you could run the rule two more times, once for US Reporting with USD, and then for CAN Reporting with CAD.

Showing the Template Flow

When you are reviewing the script generated by a system template, it is sometimes helpful to see the template flow.

The template flow shows you:

- The flow of the script that is generated by the template
- The text that the design-time prompts are replaced with by the selections you made in the steps of the template

To show the flow of a template:

1. In **System View**, right-click the business rule or custom template that contains the system template, and then select **Open**.
2. If you have not done so already, fill in all the information in the template.
3. In the Rule Designer flow chart, right-click the system template, and then select **Show Template Flow**.

   The template flow is displayed in the Template Designer in read-only mode. You can click on each component in the flow chart to display the script associated with that component and the values entered in that steps for the related DTPs.

   If a component in the flow is shown in grey, that indicates that the component is not part of the script generation based on the selections you made when entering data in the template.

   It is especially helpful to see the template flow when there is a condition component and script for both the true and false sides of the condition. The path taken by the template to generate the script is displayed in bold, and the other path is displayed in grey.

Saving a System Template as a Custom Template

You may want to customize the content of a system template.

Although you cannot edit system templates, you can use Save As on a system template to create a custom template that you can edit. The original system template is unchanged.

To save a system template as a custom template:

1. In **System View**, right-click the business rule that contains the system template, and then select **Open**.
2. Under **New Objects** or **Existing Objects**, or in the Rule Designer flow chart or script, right-click the system template, and then select **Open**.
3. When the system template opens in the **Template Designer**, it will open as read-only. Click **OK**, then select **Actions**, and then **Save As**.
4. In the **Save As** dialog box, enter a new name for the template, then select an application type, application, and cube, and then click **OK**.

   The new template is displayed in the **Templates** node of the application and cube that you selected. You can open it and customize it.

> **Note**
>
> You may need to refresh the application list in **System View** to see the new template. Right-click the **Templates** node, and the select **Refresh**.

Removing a System Template from a Business Rule

You can remove system templates from business rules.

To remove a system template:

1. In **System View**, right-click the business rule from which you want to remove the system template, and then select **Open**.
2. For a graphical rule, in the flow chart of the Rule Designer, right-click the system template, and then select **Remove**.
3. For a script rule, highlight the entire line that contains the system template, then right-click, and then select **Delete**.
4. Click  .

6

Working with Custom Templates

Related Topics

- About Custom Templates
  A custom template is designed by an administrator for use in business rules and other templates.
- Creating a Custom Template
  You can create *graphical* custom templates and *script* custom templates.
- Creating Design-Time Prompts for Custom Templates
  You can enter design-time prompts for graphical custom templates so that when you use the template to design business rules, you are prompted to enter the correct information.
- Opening a Custom Template
  You can open a custom template from System View, Deployment View, and Custom View.
- Refreshing Custom Templates
  After you create a custom template, you may need to refresh the application list in System View to see the new template in the Templates node.
- Showing the Usages of a Custom Template
  You can display a list of the business rules that are using a custom template.
- Copying and Pasting a Custom Template
  You can copy a custom template and paste it into another business rule.
- Deleting a Custom Template
  You can delete a custom template from System View, Custom View, and Deployment View.
- Finding and Replacing Text in Graphical Custom Templates
  You can search for text strings in a custom template. You can also replace all instances of a text string.

About Custom Templates

A custom template is designed by an administrator for use in business rules and other templates.

You can access custom templates from:

- The **Templates** node of a plan type within any of the views
- The Rule Designer under **Existing Objects**

There are two types of custom templates:

- A *graphical* custom template can contain formulas, script, conditions, member blocks, member ranges, fixed loops, metadata loops, and DTP assignment components.
- A *script* custom template can contain script, but no components.

Both graphical and script custom templates can also contain existing rules, formulas and scripts (shared components), other custom templates, and system templates.

You can include both the graphical and script templates in rules, or other custom templates.

Differences between graphical and script templates:

- Graphical templates have DTP assignment components, metadata loop components, and the ability to use DTP conditions within the components.
- Script templates can be deployed to and launched in Planning.

Creating a Custom Template

You can create *graphical* custom templates and *script* custom templates.

Related Topics

- Creating a Graphical Custom Template
- Creating a Script Custom Template
- Creating a Groovy Template for a Planning BSO Cube
- Creating a Groovy Template for a Planning ASO Cube

Creating a Graphical Custom Template

To create a graphical custom template:

1. In **System View**, right-click **Templates**, and then select **New**.
2. In **New Custom Defined Template**, enter the requested information, and then click **OK**.

   The **New Custom Defined Template** dialog box is automatically populated with the Planning application type, application, and plan type you are working with in the System View.
3. In the **Template Designer**, add new and existing objects to the template's flow chart.

   To add an object, drag an object from **New Objects** or **Existing Objects** and drop in into the flow chart between **Begin** and **End**.

Table 6-1 New Objects to Insert in Graphical Custom Templates

Object	See Also
Formula	Formula Components
Script	Script Components
Condition	Condition Components
Member Block	Member Block Components
Member Range	Member Range Components
Fixed Loop	Fixed Loop Components
Metadata Loop	Metadata Loop Components
DTP Assignment	DTP Assignment Components

Table 6-2 Existing Objects to Insert in Graphical Custom Templates

Object	See Also
Rules	Designing Business Rules
Scripts	Sharing Script and Formula Components
Formulas	Sharing Script and Formula Components

Table 6-2 (Cont.) Existing Objects to Insert in Graphical Custom Templates

Object	See Also
Templates	Working with Custom Templates

4. In **Properties**, enter properties for the template.

   The properties change as you add components to the template and move among the components in the flow chart. To enter properties for a specific component, select the component in the flow chart.

5. Use the **Design Time Prompt** tab to enter the following types of design-time prompts:

   - Design-time prompts that can prompt you to enter information when you are using the template.
   - Non-promptable design-time prompts that can be used in DTP assignment components, enabling conditions, and validation conditions.

   See Creating Design-Time Prompts for Custom Templates.

   When you open a custom template, by default it contains these system design-time prompts, which you can add to your template if desired:

   - **Application Type**—Used to determine if the application is a classic (generic) or Oracle Project Financial Planning (PFP) application.
   - **Application**—Used to determine whether the application is single, multi, or simplified currency.
   - **UpperPOV**—Used to determine if there are members in the global range or if the template was inserted into a member range. See Example of Creating a Graphical Template that Uses an UpperPOV Design-Time Prompt.

6. Click  to save the template.

> **Tip**
>
> As you edit the components in a template, you can increase or decrease the size of the component icons and the amount of detail that is displayed in the flow chart. To do this, use  to zoom in and out within the flow chart.

Example of Creating a Graphical Template that Uses an UpperPOV Design-Time Prompt

The following steps show you how to create a graphical template that uses an UpperPOV design-time prompt. The steps in this example use the sample Planning application.

1. In **System View**, right-click **Templates**, and then select **New**.
2. In **New Custom Defined Template**, enter the requested information, and then click **OK**.
3. In the **Design Time Prompt** tab, click  , and then select **Insert Row at End**.
4. In the **Create Prompt** dialog box, do the following:
   a. In the **Properties** tab, enter the following information:
      - **Name**—Enter "MR1".

- **Type**—Select "Member Range"
- Select **Prompt It?**
- **DTP Text**—Enter a member range

b. In the **Dependencies** tab, click the drop down for **UpperPOV**, and then select **Exclusive**.

c. Click **OK**.

5. In the **Template Designer** flow chart, drag in a *member range* component.
6. In the **Member Range** tab, click  .
7. In the **DTP Selector** dialog box, select the "Member Range" design-time prompt created in Step 4, then click **OK**, and then verify that **Link Variable Dynamically** is selected.
8. Create a step in the template, and then add a new DTP named "MR1" to the step.
9. Save the template.
10. In a rule, drag in a *member range*, and then enter members for "Entity" and "Product".
11. Drag the new graphical template into the member range in the rule.

Notice the step for the *member range* type DTP is displayed, and notice that the "Entity" and "Product" dimensions are not displayed. This is because we made the DTP of type *member range* exclusive of the UpperPOV, and the UpperPOV (the member range we dragged into the rule) is using the dimensions "Entity" and "Product".

If you change the dependency on the DTP of type *member range* to *inclusive*, in the rule, only the dimension "Entity" will be displayed, and in the template's step, only the dimension "Product" will be displayed.

You can also use the UpperPOV system DTP in a graphical template in a DTP condition to determine if the UpperPOV has members (Is Not Empty) or does not have members(Is Empty), and if there is a member range component in which the template has been dragged (Is Available), or not (Is Not Available)

Creating a Script Custom Template

When you create a custom template using script, you can:

- Deploy the script template to Planning
- Launch the template in Planning, where the steps will display allowing you to enter data at runtime
- Use run-time functions and design-time prompts

To create a custom template using a script:

1. In **System View**, right-click **Templates**, and then select **New**.
2. In the **New Custom Defined Template** dialog box, enter the requested information, and then click **OK**.

   The dialog box is automatically populated with the Planning application type, application, and cube you are working with in the System View.
3. When the template is opened, click the drop-down next to **Designer**, and then select **Edit Script**.

   This converts the template from graphical to script.

4. In the **Script Type** drop-down, select **Calc Script**.

   If you do not see the **Script Type** drop-down, click **>>** at the right of the Template Designer.

5. In **Properties**, enter properties for the template.
6. In the **Template Designer** type in the script.
7. Use the **Design Time Prompt** tab to enter design-time prompts that prompt you to enter information when you launch the template in Planning. See Creating Design-Time Prompts for Custom Templates
8. Click  to save the template.
9. Deploy the template to Planning.
10. Open Planning, and then launch the template.

Creating a Groovy Template for a Planning BSO Cube

To create a Groovy template for a Planning BSO cube:

1. In **System View**, right-click **Templates**, and then select **New**.
2. In the **New Custom Defined Template** dialog box, enter the requested information, and then click **OK**.

   The dialog box is automatically populated with the Planning application type, application, and cube you are working with in the System View.

3. When the template is opened, click the drop-down next to **Designer**, and then select **Edit Script**.
4. In the **Script Type** drop-down, select **Groovy Script**.

   If you do not see the **Script Type** drop-down, click **>>** at the right of the Template Designer.

5. In **Properties**, enter properties for the template.
6. In the **Template Designer** type in the script.
7. Use the **Design Time Prompt** tab to enter design-time prompts that prompt you to enter information when you launch the template in Planning.

   See Creating Design-Time Prompts for Custom Templates.

8. Click  to save the template.
9. Deploy the template to Planning.
10. Open Planning, and then launch the template.

> **Note**
>
> See Assigning Access to Groovy Templates for information on how to assign access to Groovy templates in Planning.

Creating a Groovy Template for a Planning ASO Cube

To create a Groovy template for a Planning ASO cube:

1. In **System View**, right-click **Templates**, and then select **New**.
2. In the **New Custom Defined Template** dialog box, enter the requested information, and then click **OK**.

   The dialog box is automatically populated with the Planning application type, application, and cube you are working with in the System View.
3. In **Properties**, enter properties for the template.
4. In the **Template Designer** type in the script.
5. Use the **Design Time Prompt** tab to enter design-time prompts that prompt you to enter information when you launch the template in Planning.

   See Creating Design-Time Prompts for Custom Templates.
6. Click  to save the template.
7. Deploy the template to Planning.
8. Open Planning, and then launch the template.

Creating Design-Time Prompts for Custom Templates

You can enter design-time prompts for graphical custom templates so that when you use the template to design business rules, you are prompted to enter the correct information.

Related Topics

- Types of Design-Time Prompts
- Creating Steps for Design-Time Prompts
- Defining Dependencies for Design-Time Prompts
- Defining Limits for Design-time Prompts
- Finding and Replacing Text in Design-Time Prompts

Types of Design-Time Prompts

Attribute DTP

Defines an attribute from the application to which the custom template belongs.

For example, assume you create a design-time prompt to enter the size of a product. In this case, you could select a "Size" dimension. You could also enter a default value if desired; for example, "Large". If you select **Prompt It?**, then when the template is used, the user is prompted to enter an attribute member (in this example, a dimension size).

To create an Attribute design-time prompt:

1. In the **Template Designer**, open an existing template or create a new template.
2. On the **Design Time Prompt** tab, click  .

3. If you already have design-time prompts defined for the template, select where to enter the new design-time prompt.
4. In the **Create Prompt** dialog box, on the **Properties** tab, do the following:
   - Enter a **Name** and select **Attribute** as the **Type**.
   - Select the desired options:
     - **Prompt It?**—Select this option to allow the DTP to be used in a step. The DTP is shown in the template steps that prompt users for input when the template is used in a rule, or when a script template is launched in Planning.
     - **Mandatory?**—Select this option to make it mandatory to enter data for the design-time prompt.
     - **Read Only**—Select this option to make it so that users can only read the prompt.
     - **Is Hidden**—Select this option to hide the design-time prompt from the user. In Planning, on the **Business Rules** tab in form design, add the rule with the template or script template to the form. In the **Business Rules Properties** pane, select **Hide Prompt**, and then select **Use Members on Form**.
   - In **DTP Text**, enter the text that you want users to see when they are prompted for input.
     If you select **Prompt It?**, then **DTP Text** is required.
   - **Optional.** In **Comments**, enter a comment about the DTP.
   - In **Dimension**, select an attribute dimension.

     The Dimension drop-down displays the attributes from the current application. If you do not select a dimension, when you are prompted for input, you will see all the attribute dimensions.
   - **Optional.** In **Default Value**, assign a default value for the DTP.

     To assign a default value, click  , then select **Member**, and then select a member in the **Member Selector**.

     The attribute dimension selected in the **Dimension** drop-down is displayed. Expand this dimension to display the available members.

Boolean DTP

"True" or "False" type DTP.

To create an Boolean design-time prompt:

1. In the **Template Designer**, open an existing template or create a new template.
2. On the **Design Time Prompt** tab, click  .
3. If you already have design-time prompts defined for the template, select where to enter the new design-time prompt.
4. In the **Create Prompt** dialog box, on the **Properties** tab, do the following:
   - Enter a **Name** and select **Boolean** as the **Type**.
   - Select the desired options:
     - **Prompt It?**—Select this option to allow the DTP to be used in a step. The DTP is shown in the template steps that prompt users for input when the template is used in a rule, or when a script template is launched in Planning.

  - **Mandatory?**—Select this option to make it mandatory to enter data for the design-time prompt.
  - **Read Only**—Select this option to make it so that users can only read the prompt.
  - **Is Hidden**—Select this option to hide the design-time prompt from the user. In Planning, on the **Business Rules** tab in form design, add the rule with the template or script template to the form. In the **Business Rules Properties** pane, select **Hide Prompt**, and then select **Use Members on Form**.
- In **DTP Text**, enter the text that you want users to see when they are prompted for input.
  If you select **Prompt It?**, then **DTP Text** is required.
- **Optional.** In **Comments**, enter a comment about the DTP.
- **Optional.** In **Default Value**, select **True** or **False**.

Cross Dimension DTP

Defines a cross-dimension syntax (->) range of members from the application to which the template belongs.

A Cross Dimension DTP allows users to select one member from each dimension. For example, a cross Dimension design-time prompt could be used in the source of a formula.

1. In the **Template Designer**, open an existing template or create a new template.
2. On the **Design Time Prompt** tab, click  .
3. If you already have design-time prompts defined for the template, select where to enter the new design-time prompt.
4. In the **Create Prompt** dialog box, on the **Properties** tab, do the following:
   - Enter a **Name** and select **Cross Dimension** as the **Type**.
   - Select the desired options:
     - **Prompt It?**—Select this option to allow the DTP to be used in a step. The DTP is shown in the template steps that prompt users for input when the template is used in a rule, or when a script template is launched in Planning.
     - **Mandatory?**—Select this option to make it mandatory to enter data for the design-time prompt.
     - **Read Only**—Select this option to make it so that users can only read the prompt.
     - **Is Hidden**—Select this option to hide the design-time prompt from the user. In Planning, on the **Business Rules** tab in form design, add the rule with the template or script template to the form. In the **Business Rules Properties** pane, select **Hide Prompt**, and then select **Use Members on Form**.
   - In **DTP Text**, enter the text that you want users to see when they are prompted for input.
     If you select **Prompt It?**, then **DTP Text** is required.
   - **Optional.** In **Comments**, enter a comment about the DTP.
   - **Optional.** In **Default Value**, assign a default value for the DTP.

     To assign a default value, do one of the following:

     - Click  , then click  next to each dimension, and then select **DTP**, **Member**, or **Function**.

  - Click  to select a DTP as the default value.

5. **Optional:** On the **Dependencies** tab, select a design-time prompt, and then select a dependency option:
   - **Exclusive**—Makes the design-time prompt (for example, DTP2) exclusive of another design-time prompt (for example DTP1). When you are prompted for DTP2, only the dimensions that are not used in DTP1 are displayed.
   - **Inclusive**—Makes the design-time prompt (for example, DTP4) inclusive of another design-time prompt (for example, DTP3). When you are prompted for DTP4, only the dimensions that are used in DTP3 are displayed.
   - **None**—No dependencies are defined for the design-time prompt.
6. **Optional:** On the **Define Limits** tab, do the following:
   - In **Density**, select an option:
     - **Show Sparse**—Displays only sparse dimensions for the DTP
     - **Show Dense**—Displays only dense dimensions for the DTP
     - **Both**—Displays both dense and sparse dimensions for the DTP
   - In **Dimension Type** select a dimension type to restrict the DPT to show only the selected dimension types.

     For example, if you only want users to enter a member for *Account*, *Entity*, and *Year*, then select these dimensions in **Dimension Type**. When users are prompted for input, the only dimensions that will show are those with type *Account*, *Entity*, and *Year*.

Condition DTP

Defines a condition.

To create an Condition design-time prompt:

1. In the **Template Designer**, open an existing template or create a new template.
2. On the **Design Time Prompt** tab, click  .
3. If you already have design-time prompts defined for the template, select where to enter the new design-time prompt.
4. In the **Create Prompt** dialog box, on the **Properties** tab, do the following:
   - Enter a **Name** and select **Condition** as the **Type**.
   - Select the desired options:
     - **Prompt It?**—Select this option to allow the DTP to be used in a step. The DTP is shown in the template steps that prompt users for input when the template is used in a rule, or when a script template is launched in Planning.
     - **Mandatory?**—Select this option to make it mandatory to enter data for the design-time prompt.
     - **Read Only**—Select this option to make it so that users can only read the prompt.
     - **Is Hidden**—Select this option to hide the design-time prompt from the user. In Planning, on the **Business Rules** tab in form design, add the rule with the template or script template to the form. In the **Business Rules Properties** pane, select **Hide Prompt**, and then select **Use Members on Form**.

- In **DTP Text**, enter the text that you want users to see when they are prompted for input.

  If you select **Prompt It?**, then **DTP Text** is required.
- **Optional.** In **Comments**, enter a comment about the DTP.

DateAsNumber DTP

Defines a date as a number.

To create a DateAsNumber design-time prompt:

1. In the **Template Designer**, open an existing template or create a new template.
2. On the **Design Time Prompt** tab, click \  .
3. If you already have design-time prompts defined for the template, select where to enter the new design-time prompt.
4. In the **Create Prompt** dialog box, on the **Properties** tab, do the following:
   - Enter a **Name** and select **DateAsNumber** as the **Type**.
   - Select the desired options:
     - **Prompt It?**—Select this option to allow the DTP to be used in a step. The DTP is shown in the template steps that prompt users for input when the template is used in a rule, or when a script template is launched in Planning.
     - **Mandatory?**—Select this option to make it mandatory to enter data for the design-time prompt.
     - **Read Only**—Select this option to make it so that users can only read the prompt.
     - **Is Hidden**—Select this option to hide the design-time prompt from the user. In Planning, on the **Business Rules** tab in form design, add the rule with the template or script template to the form. In the **Business Rules Properties** pane, select **Hide Prompt**, and then select **Use Members on Form**.
   - In **DTP Text**, enter the text that you want users to see when they are prompted for input.

     If you select **Prompt It?**, then **DTP Text** is required.
   - **Optional.** In **Comments**, enter a comment about the DTP.
   - Select **Allow #Missing** to allow #Missing values in the DTP instead of a date.
   - In **Limits**, click  , and then define the minimum and maximum number you can enter in the design-time prompt.
   - **Optional.** In **Default Value**, click  , then select **DTP**, and then select a DTP value.

Dimension DTP

Defines a dimension from the application to which the template belongs.

To create an Dimension design-time prompt:

1. In the **Template Designer**, open an existing template or create a new template.
2. On the **Design Time Prompt** tab, click  .

3. If you already have design-time prompts defined for the template, select where to enter the new design-time prompt.
4. In the **Create Prompt** dialog box, on the **Properties** tab, do the following:
   - Enter a **Name** and select **Dimension** as the **Type**.
   - Select the desired options:
     - **Prompt It?**—Select this option to add the DTP to a step. This DTP is shown in the template steps that prompt users for input when the template is used in a rule, or when a script template is launched in Planning.
     - **Mandatory?**—Select this option to make it mandatory to enter data for the design-time prompt.
     - **Read Only**—Select this option to make it so that users can only read the prompt.
     - **Is Hidden**—Select this option to hide the design-time prompt from the user. In Planning, on the **Business Rules** tab in form design, add the rule with the template or script template to the form. In the **Business Rules Properties** pane, select **Hide Prompt**, and then select **Use Members on Form**.
   - In **DTP Text**, enter the text that you want users to see when they are prompted for input.

     If you select **Prompt It?**, then **DTP Text** is required.
   - **Optional.** In **Comments**, enter a comment about the DTP.
   - **Optional.** In **Default Value**, click the drop-down, and then select a dimension from the application to which the template belongs.
5. **Optional.** On the **Dependencies** tab, select a design-time prompt, and then select a dependency option:
   - **Exclusive**—Makes the design-time prompt (for example, DTP2) exclusive of another design-time prompt (for example DTP1). When you are prompted for DTP2, only the dimensions that are not used in DTP1 are displayed.
   - **Inclusive**—Makes the design-time prompt (for example, DTP4) inclusive of another design-time prompt (for example, DTP3). When you are prompted for DTP4, only the dimensions that are used in DTP3 are displayed.
   - **None**—No dependencies are defined for the design-time prompt.

   Dimension DTPs can have dependencies on Member Range, Dimension, Dimensions, or Cross Dimension DTPs.
6. **Optional. On** the **Define Limits** tab, do the following:
   - In **Density**, select an option:
     - **Show Sparse**—Displays only sparse dimensions for the DTP
     - **Show Dense**—Displays only dense dimensions for the DTP
     - **Both**—Displays both dense and sparse dimensions for the DTP
   - In **Dimension Type** select a dimension type to restrict the DPT to show only the selected dimension types.

     For example, if you only want users to enter a member for *Account*, *Entity*, and *Year*, then select these dimensions in **Dimension Type**. When users are prompted for input, the only dimensions that will show are those with type *Account*, *Entity*, and *Year*.

Dimensions DTP

Defines dimensions from the application to which the template belongs.

To create an Dimensions design-time prompt:

1. In the **Template Designer**, open an existing template or create a new template.
2. On the **Design Time Prompt** tab, click  .
3. If you already have design-time prompts defined for the template, select where to enter the new design-time prompt.
4. In the **Create Prompt** dialog box, on the **Properties** tab, do the following:
   - Enter a **Name** and select **Dimension** as the **Type**.
   - Select the desired options:
     - **Prompt It?**—Select this option to allow the DTP to be used in a step. The DTP is shown in the template steps that prompt users for input when the template is used in a rule, or when a script template is launched in Planning.
     - **Mandatory?**—Select this option to make it mandatory to enter data for the design-time prompt.
     - **Read Only**—Select this option to make it so that users can only read the prompt.
     - **Is Hidden**—Select this option to hide the design-time prompt from the user. In Planning, on the **Business Rules** tab in form design, add the rule with the template or script template to the form. In the **Business Rules Properties** pane, select **Hide Prompt**, and then select **Use Members on Form**.
   - In **DTP Text**, enter the text that you want users to see when they are prompted for input.
     If you select **Prompt It?**, then **DTP Text** is required.
   - **Optional.** In **Comments**, enter a comment about the DTP.
   - **Optional.** In **Default Value**, assign a default value for the DTP.

     To assign a default value, click  , then select **DTP** or **Dimensions**, and then select a DTP or a dimension.
5. **Optional.** On the **Dependencies** tab, select a design-time prompt, and then select a dependency option:
   - **Exclusive**—Makes the design-time prompt (for example, DTP2) exclusive of another design-time prompt (for example DTP1). When you are prompted for DTP2, only the dimensions that are not used in DTP1 are displayed.
   - **Inclusive**—Makes the design-time prompt (for example, DTP4) inclusive of another design-time prompt (for example, DTP3). When you are prompted for DTP4, only the dimensions that are used in DTP3 are displayed.
   - **None**—No dependencies are defined for the design-time prompt.

   Dimensions DTPs can have dependencies on Member Range, Dimension, Dimensions, or Cross Dimension DTPs.
6. **Optional.** On the **Define Limits** tab, do the following:
   - In **Density**, select an option:
     - **Show Sparse**—Displays only sparse dimensions for the DTP

  - **Show Dense**—Displays only dense dimensions for the DTP
  - **Both**—Displays both dense and sparse dimensions for the DTP
- In **Dimension Type** select a dimension type to restrict the DPT to show only the selected dimension types.

  For example, if you only want users to enter a member for *Account*, *Entity*, and *Year*, then select these dimensions in **Dimension Type**. When users are prompted for input, the only dimensions that will show are those with type *Account*, *Entity*, and *Year*.

Integer DTP

Defines an integer.

To create an Integer design-time prompt:

1. In the **Template Designer**, open an existing template or create a new template.
2. On the **Design Time Prompt** tab, click  .
3. If you already have design-time prompts defined for the template, select where to enter the new design-time prompt.
4. In the **Create Prompt** dialog box, on the **Properties** tab, do the following:
   - Enter a **Name** and select **Numeric** as the **Type**.
   - Select the desired options:
     - **Prompt It?**—Select this option to allow the DTP to be used in a step. The DTP is shown in the template steps that prompt users for input when the template is used in a rule, or when a script template is launched in Planning.
     - **Mandatory?**—Select this option to make it mandatory to enter data for the design-time prompt.
     - **Read Only**—Select this option to make it so that users can only read the prompt.
     - **Is Hidden**—Select this option to hide the design-time prompt from the user. In Planning, on the **Business Rules** tab in form design, add the rule with the template or script template to the form. In the **Business Rules Properties** pane, select **Hide Prompt**, and then select **Use Members on Form**.
   - In **DTP Text**, enter the text that you want users to see when they are prompted for input.
     If you select **Prompt It?**, then **DTP Text** is required.
   - **Optional.** In **Comments**, enter a comment about the DTP.
   - Select **Allow #Missing** to allow #Missing values in the DTP instead of an integer.
   - In **Limits**, click  , and then define the minimum and maximum number you can enter in the design-time prompt.
   - **Optional.** In **Default Value**, click  , then select **DTP**, and then select a DTP value.

Member DTP

Define a member from a dimension in the application.

To create a Member design-time prompt:

1. In the **Template Designer**, open an existing template or create a new template.

2. On the **Design Time Prompt** tab, click  .
3. If you already have design-time prompts defined for the template, select where to enter the new design-time prompt.
4. In the **Create Prompt** dialog box, on the **Properties** tab, do the following:
   - Enter a **Name** and select **Member** as the **Type**.
   - Select the desired options:
     - **Prompt It?**—Select this option to allow the DTP to be used in a step. The DTP is shown in the template steps that prompt users for input when the template is used in a rule, or when a script template is launched in Planning.
     - **Mandatory?**—Select this option to make it mandatory to enter data for the design-time prompt.
     - **Read Only**—Select this option to make it so that users can only read the prompt.
     - **Is Hidden**—Select this option to hide the design-time prompt from the user. In Planning, on the **Business Rules** tab in form design, add the rule with the template or script template to the form. In the **Business Rules Properties** pane, select **Hide Prompt**, and then select **Use Members on Form**.
   - In **DTP Text**, enter the text that you want users to see when they are prompted for input.

     If you select **Prompt It?**, then **DTP Text** is required.
   - **Optional.** In **Comments**, enter a comment about the DTP.
   - In **Dimension**, select a dimension.

     The Dimension drop-down displays the dimensions from the current application.

     In *graphical* templates, selecting a dimension is optional. If you select a dimension, the **Define Limits** tab is not available.

     In *script* templates, you must select a dimension.
   - **Optional.** In **Default Value**, assign a default value for the DTP.

     To assign a default value, click  , then select **Member**, and then select a member in the **Member Selector**. The attribute dimension selected in the **Dimension** drop-down is displayed. Expand this dimension to display the available members.
   - **Script templates only.** In **Limits**, enter a DTP, members or function that return a set of members from which users can select.

     For example, to limit the DTP when prompted to only show the level 0 accounts under the parent "Revenues", select "Account " as the **Dimension**; then, in **Limits**, enter "ILvL0Descendants(Revenues)".

     Note: When using a function for the Limit, Planning functions are recommended instead of Oracle Essbase functions. In some cases, Essbase functions do not return the expected members; for example, when the evaluation of the Essbase function includes dynamic members. For the Limit in a Member design-time prompt, use "ILvl0Descendants("Mbr Name") " instead of the Essbase function "@Relative("Mbr Name", 0)".
5. **Optional.** On the **Dependencies** tab, select a design-time prompt, and then select a dependency option:

- **Exclusive**—Makes the design-time prompt (for example, DTP2) exclusive of another design-time prompt (for example DTP1). When you are prompted for DTP2, only the dimensions that are not used in DTP1 are displayed.
- **Inclusive**—Makes the design-time prompt (for example, DTP4) inclusive of another design-time prompt (for example, DTP3). When you are prompted for DTP4, only the dimensions that are used in DTP3 are displayed.
- **None**—No dependencies are defined for the design-time prompt.

6. **Optional.** The **Define Limits** tab applies only to *graphical* templates, and it is displayed only if you did not select a dimension. If you are in a graphical template and did not select a dimension, do the following:
   - In **Density**, select an option:
     - **Show Sparse**—Displays only sparse dimensions for the DTP
     - **Show Dense**—Displays only dense dimensions for the DTP
     - **Both**—Displays both dense and sparse dimensions for the DTP
   - In **Dimension Type** select a dimension type to restrict the DPT to show only the selected dimension types.

     For example, if you only want users to enter a member for *Account*, *Entity*, and *Year*, then select these dimensions in **Dimension Type**. When users are prompted for input, the only dimensions that will show are those with type *Account*, *Entity*, and *Year*.

Members DTP

Defines multiple members from a selected dimension in the application.

To create an Members design-time prompt:

1. In the **Template Designer**, open an existing template or create a new template.
2. On the **Design Time Prompt** tab, click  .
3. If you already have design-time prompts defined for the template, select where to enter the new design-time prompt.
4. In the **Create Prompt** dialog box, on the **Properties** tab, do the following:
   - Enter a **Name** and select **Members** as the **Type**.
   - Select the desired options:
     - **Prompt It?**—Select this option to allow the DTP to be used in a step. The DTP is shown in the template steps that prompt users for input when the template is used in a rule, or when a script template is launched in Planning.
     - **Mandatory?**—Select this option to make it mandatory to enter data for the design-time prompt.
     - **Read Only**—Select this option to make it so that users can only read the prompt.
     - **Is Hidden**—Select this option to hide the design-time prompt from the user. In Planning, on the **Business Rules** tab in form design, add the rule with the template or script template to the form. In the **Business Rules Properties** pane, select **Hide Prompt**, and then select **Use Members on Form**.
   - In **DTP Text**, enter the text that you want users to see when they are prompted for input.

     If you select **Prompt It?**, then **DTP Text** is required.

- **Optional.** In **Comments**, enter a comment about the DTP.
- In **Dimension**, select a dimension.

  The Dimension drop-down displays the dimensions from the current application.

  In *graphical* templates, selecting a dimension is optional. If you select a dimension, the **Define Limits** tab is not available.

  In *script* templates, you must select a dimension.
- **Optional.** In **Default Value**, assign a default value for the DTP.

  To assign a default value, click  , then select **Member**, and then select a member in the **Member Selector**. The attribute dimension selected in the **Dimension** drop-down is displayed. Expand this dimension to display the available members.
- **Script templates only.** In **Limits**, enter a DTP, members or function that return a set of members from which users can select.

  For example, to limit the DTP when prompted to only show the level 0 accounts under the parent "Revenues", select "Account " as the **Dimension**; then, in **Limits**, enter "ILvl0Descendants(Revenues)".

  Note: When using a function for the Limit, Planning functions are recommended instead of Oracle Essbase functions. In some cases, Essbase functions do not return the expected members; for example, when the evaluation of the Essbase function includes dynamic members. For the Limit in a Members design-time prompt, use "ILvl0Descendants("Mbr Name") " instead of the Essbase function "@Relative("Mbr Name", 0)".

5. **Optional.** On the **Dependencies** tab, select a design-time prompt, and then select a dependency option:
   - **Exclusive**—Makes the design-time prompt (for example, DTP2) exclusive of another design-time prompt (for example DTP1). When you are prompted for DTP2, only the dimensions that are not used in DTP1 are displayed.
   - **Inclusive**—Makes the design-time prompt (for example, DTP4) inclusive of another design-time prompt (for example, DTP3). When you are prompted for DTP4, only the dimensions that are used in DTP3 are displayed.
   - **None**—No dependencies are defined for the design-time prompt.
6. **Optional.** The **Define Limits** tab applies only to *graphical* templates, and it is displayed only if you did not select a dimension. If you are in a graphical template and did not select a dimension, do the following:
   - In **Density**, select an option:
     - **Show Sparse**—Displays only sparse dimensions for the DTP
     - **Show Dense**—Displays only dense dimensions for the DTP
     - **Both**—Displays both dense and sparse dimensions for the DTP
   - In **Dimension Type** select a dimension type to restrict the DPT to show only the selected dimension types.

     For example, if you only want users to enter a member for *Account*, *Entity*, and *Year*, then select these dimensions in **Dimension Type**. When users are prompted for input, the only dimensions that will show are those with type *Account*, *Entity*, and *Year*.

Member Range DTP

Defines a range of members from selected dimensions in the application.

A Member Range DTP can have one or multiple members from each dimension. For example, you could use a Member Range type DTP to create a Point of View step where you ask users to input members for a rule.

To create an Member Range design-time prompt:

1. In the **Template Designer**, open an existing template or create a new template.
2. On the **Design Time Prompt** tab, click  .
3. If you already have design-time prompts defined for the template, select where to enter the new design-time prompt.
4. In the **Create Prompt** dialog box, on the **Properties** tab, do the following:
   - Enter a **Name** and select **Member Range** as the **Type**.
   - Select the desired options:
     - **Prompt It?**—Select this option to allow the DTP to be used in a step. The DTP is shown in the template steps that prompt users for input when the template is used in a rule, or when a script template is launched in Planning.
     - **Mandatory?**—Select this option to make it mandatory to enter data for the design-time prompt.
     - **Read Only**—Select this option to make it so that users can only read the prompt.
     - **Is Hidden**—Select this option to hide the design-time prompt from the user. In Planning, on the **Business Rules** tab in form design, add the rule with the template or script template to the form. In the **Business Rules Properties** pane, select **Hide Prompt**, and then select **Use Members on Form**.
   - In **DTP Text**, enter the text that you want users to see when they are prompted for input.

     If you select **Prompt It?**, then **DTP Text** is required.
   - **Optional.** In **Comments**, enter a comment about the DTP.
   - **Optional.** In **Default Value**, assign a default value for the DTP.

     To assign a default value, do one of the following:
     - Click  , then click  next to each dimension, and then select **DTP**, **Member**, or **Function**.
     - Click  to use a DTP as the default value.
5. **Optional.** On the **Dependencies** tab, select a design-time prompt, and then select a dependency option:
   - **Exclusive**—Makes the design-time prompt (for example, DTP2) exclusive of another design-time prompt (for example DTP1). When you are prompted for DTP2, only the dimensions that are not used in DTP1 are displayed.
   - **Inclusive**—Makes the design-time prompt (for example, DTP4) inclusive of another design-time prompt (for example, DTP3). When you are prompted for DTP4, only the dimensions that are used in DTP3 are displayed.

- **None**—No dependencies are defined for the design-time prompt.

Member Range DTPs can be dependent on the following types of DTPs: Member Range, Cross Dimension, Dimension, and Dimensoons.

6. **Optional.** On the **Define Limits** tab, do the following:
   - In **Density**, select an option:
     - **Show Sparse**—Displays only sparse dimensions for the DTP
     - **Show Dense**—Displays only dense dimensions for the DTP
     - **Both**—Displays both dense and sparse dimensions for the DTP
   - In **Dimension Type** select a dimension type to restrict the DPT to show only the selected dimension types.

     For example, if you only want users to enter a member for *Account*, *Entity*, and *Year*, then select these dimensions in **Dimension Type**. When users are prompted for input, the only dimensions that will show are those with type *Account*, *Entity*, and *Year*.

Numeric DTP

Defines a number. For example, you could use a Numeric DTP in a formula.

To create an Numeric design-time prompt:

1. In the **Template Designer**, open an existing template or create a new template.
2. On the **Design Time Prompt** tab, click  .
3. If you already have design-time prompts defined for the template, select where to enter the new design-time prompt.
4. In the **Create Prompt** dialog box, on the **Properties** tab, do the following:
   - Enter a **Name** and select **Numeric** as the **Type**.
   - Select the desired options:
     - **Prompt It?**—Select this option to allow the DTP to be used in a step. The DTP is shown in the template steps that prompt users for input when the template is used in a rule, or when a script template is launched in Planning.
     - **Mandatory?**—Select this option to make it mandatory to enter data for the design-time prompt.
     - **Read Only**—Select this option to make it so that users can only read the prompt.
     - **Is Hidden**—Select this option to hide the design-time prompt from the user. In Planning, on the **Business Rules** tab in form design, add the rule with the template or script template to the form. In the **Business Rules Properties** pane, select **Hide Prompt**, and then select **Use Members on Form**.
   - In **DTP Text**, enter the text that you want users to see when they are prompted for input.

     If you select **Prompt It?**, then **DTP Text** is required.
   - **Optional.** In **Comments**, enter a comment about the DTP.
   - Select **Allow #Missing** to allow #Missing values in the DTP instead of a number.
   - In **Limits**, click  , and then define the minimum and maximum number you can enter in the design-time prompt.

- **Optional.** In **Default Value**, click , then select **DTP**, and then select a DTP value.

Password DTP

Defines a password to use in the design-time prompt.

To create a Password design-time prompt:

1. In the **Template Designer**, open an existing template or create a new template.
2. On the **Design Time Prompt** tab, click .
3. If you already have design-time prompts defined for the template, select where to enter the new design-time prompt.
4. In the **Create Prompt** dialog box, on the **Properties** tab, do the following:
   - Enter a **Name** and select **Password** as the **Type**.
   - Select the desired options:
     - **Prompt It?**—Select this option to allow the DTP to be used in a step. The DTP is shown in the template steps that prompt users for input when the template is used in a rule, or when a script template is launched in Planning.
     - **Mandatory?**—Select this option to make it mandatory to enter data for the design-time prompt.
     - **Read Only**—Select this option to make it so that users can only read the prompt.
     - **Is Hidden**—Select this option to hide the design-time prompt from the user. In Planning, on the **Business Rules** tab in form design, add the rule with the template or script template to the form. In the **Business Rules Properties** pane, select **Hide Prompt**, and then select **Use Members on Form**.
   - In **DTP Text**, enter the text that you want users to see when they are prompted for input.

     If you select **Prompt It?**, then **DTP Text** is required.
   - **Optional.** In **Comments**, enter a comment about the DTP.
   - **Optional.** In **Default Value**, type in the value for the password.

Percent DTP

Defines a percentage.

To create an Percent design-time prompt:

1. In the **Template Designer**, open an existing template or create a new template.
2. On the **Design Time Prompt** tab, click .
3. If you already have design-time prompts defined for the template, select where to enter the new design-time prompt.
4. In the **Create Prompt** dialog box, on the **Properties** tab, do the following:
   - Enter a **Name** and select **Percent** as the **Type**.
   - Select the desired options:

  - **Prompt It?**—Select this option to allow the DTP to be used in a step. The DTP is shown in the template steps that prompt users for input when the template is used in a rule, or when a script template is launched in Planning.
  - **Mandatory?**—Select this option to make it mandatory to enter data for the design-time prompt.
  - **Read Only**—Select this option to make it so that users can only read the prompt.
  - **Is Hidden**—Select this option to hide the design-time prompt from the user. In Planning, on the **Business Rules** tab in form design, add the rule with the template or script template to the form. In the **Business Rules Properties** pane, select **Hide Prompt**, and then select **Use Members on Form**.
- In **DTP Text**, enter the text that you want users to see when they are prompted for input.

  If you select **Prompt It?**, then **DTP Text** is required.
- **Optional.** In **Comments**, enter a comment about the DTP.
- Select **Allow #Missing** to allow #Missing values in the DTP instead of a percentage.
- In **Limits**, click  , and then define the minimum and maximum number you can enter in the design-time prompt.
- **Optional.** In **Default Value**, click  , then select **DTP**, and then select a DTP value.

Restricted List DTP

Defines a restricted list.

For example, in the Allocation system template, you can select a rounding method and then use a Restricted List design-time prompt to choose the type of rounding to use in the template.

In the script of the template, you can create script for each type of rounding, and the script is only used when the user selects that rounding option.

To create a Restricted List design-time prompt:

1. In the **Template Designer**, open an existing template or create a new template.
2. On the **Design Time Prompt** tab, click  .
3. If you already have design-time prompts defined for the template, select where to enter the new design-time prompt.
4. In the **Create Prompt** dialog box, on the **Properties** tab, do the following:
   - Enter a **Name** and select **Restricted List** as the **Type**.
   - Select the desired options:
     - **Prompt It?**—Select this option to allow the DTP to be used in a step. The DTP is shown in the template steps that prompt users for input when the template is used in a rule, or when a script template is launched in Planning.
     - **Mandatory?**—Select this option to make it mandatory to enter data for the design-time prompt.
     - **Read Only**—Select this option to make it so that users can only read the prompt.
     - **Is Hidden**—Select this option to hide the design-time prompt from the user. In Planning, on the **Business Rules** tab in form design, add the rule with the

template or script template to the form. In the **Business Rules Properties** pane, select **Hide Prompt**, and then select **Use Members on Form**.

- In **DTP Text**, enter the text that you want users to see when they are prompted for input.

  If you select **Prompt It?**, then **DTP Text** is required.

- **Optional.** In **Comments**, enter a comment about the DTP.

5. Select the **Restricted List** tab, and enter a **Rule Builder Value** and a **Substituted Value**.
6. **Optional.** Go back to the **Properties** tab and enter a default value to be displayed in the design-time prompt.

   To define a default value, click the **Default Value** drop down, and then select a **Rule Builder Value** that you entered on the **Restricted List** tab.

Separator DTP

A separator is an instruction or a spacer in the template step.

To create a Separator design-time prompt:

1. In the **Template Designer**, open an existing template or create a new template.
2. On the **Design Time Prompt** tab, click  .
3. If you already have design-time prompts defined for the template, select where to enter the new design-time prompt.
4. In the **Create Prompt** dialog box, on the **Properties** tab, do the following:
   - Enter a **Name** and select **Separator** as the **Type**.
   - Select the desired options:
     - **Prompt It?**—Select this option to allow the DTP to be used in a step. The DTP is shown in the template steps that prompt users for input when the template is used in a rule, or when a script template is launched in Planning.
     - **Mandatory?**—Select this option to make it mandatory to enter data for the design-time prompt.
     - **Read Only**—Select this option to make it so that users can only read the prompt.
     - **Is Hidden**—Select this option to hide the design-time prompt from the user. In Planning, on the **Business Rules** tab in form design, add the rule with the template or script template to the form. In the **Business Rules Properties** pane, select **Hide Prompt**, and then select **Use Members on Form**.
   - In **DTP Text**, enter the text that you want users to see when they are prompted for input.

     If you select **Prompt It?**, then **DTP Text** is required.
   - **Optional.** In **Comments**, enter a comment about the DTP.

Smart List DTP

Defines a Smart List to use in the design-time prompt.

To create a Smart List design-time prompt:

1. In the **Template Designer**, open an existing template or create a new template.

2. On the **Design Time Prompt** tab, click  .
3. If you already have design-time prompts defined for the template, select where to enter the new design-time prompt.
4. In the **Create Prompt** dialog box, on the **Properties** tab, do the following:
   - Enter a **Name** and select **Smart List** as the **Type**.
   - Select the desired options:
     - **Prompt It?**—Select this option to allow the DTP to be used in a step. The DTP is shown in the template steps that prompt users for input when the template is used in a rule, or when a script template is launched in Planning.
     - **Mandatory?**—Select this option to make it mandatory to enter data for the design-time prompt.
     - **Read Only**—Select this option to make it so that users can only read the prompt.
     - **Is Hidden**—Select this option to hide the design-time prompt from the user. In Planning, on the **Business Rules** tab in form design, add the rule with the template or script template to the form. In the **Business Rules Properties** pane, select **Hide Prompt**, and then select **Use Members on Form**.
   - In **DTP Text**, enter the text that you want users to see when they are prompted for input.

     If you select **Prompt It?**, then **DTP Text** is required.
   - **Optional.** In **Comments**, enter a comment about the DTP.
5. In **Smart List**, click the drop-down, and then select a Smart List.

   The Smart List that are displayed are populated from Planning. If there are no Smart Lists in Planning, then the drop-down is empty.
6. **Optional.** In **Default Value**, click the drop down, and then select a default value.

StringAsNumber DTP

Defines a string as a number.

To create a StringAsNumber design-time prompt:

1. In the **Template Designer**, open an existing template or create a new template.
2. On the **Design Time Prompt** tab, click  .
3. If you already have design-time prompts defined for the template, select where to enter the new design-time prompt.
4. In the **Create Prompt** dialog box, on the **Properties** tab, do the following:
   - Enter a **Name** and select **StringAsNumber** as the **Type**.
   - Select the desired options:
     - **Prompt It?**—Select this option to allow the DTP to be used in a step. The DTP is shown in the template steps that prompt users for input when the template is used in a rule, or when a script template is launched in Planning.
     - **Mandatory?**—Select this option to make it mandatory to enter data for the design-time prompt.
     - **Read Only**—Select this option to make it so that users can only read the prompt.

  - **Is Hidden**—Select this option to hide the design-time prompt from the user. In Planning, on the **Business Rules** tab in form design, add the rule with the template or script template to the form. In the **Business Rules Properties** pane, select **Hide Prompt**, and then select **Use Members on Form**.
- In **DTP Text**, enter the text that you want users to see when they are prompted for input.

  If you select **Prompt It?**, then **DTP Text** is required.
- **Optional.** In **Comments**, enter a comment about the DTP.
- Select **Allow #Missing** to allow #Missing values in the DTP instead of a string.
- **Optional.** Select **Use System Defaults**, and then in **Default Value**, click the drop-down to select a default value to use in the prompt. The default value is initially populated with the user name.

> **Note**
>
> If you select **Use System Defaults**, and from the default, select "user name", then you must select **Prompt It**, and enter text in **DTP Text**. Add this design-time prompt to a step. In a rule, when you are prompted for the input for this design-time prompt, select a StringAsNumber variable, that has **Use System Defaults**, and a user name as the default value.

String DTP

Defines a text string.

To create a String design-time prompt:

1. In the **Template Designer**, open an existing template or create a new template.
2. On the **Design Time Prompt** tab, click  .
3. If you already have design-time prompts defined for the template, select where to enter the new design-time prompt.
4. In the **Create Prompt** dialog box, on the **Properties** tab, do the following:
   - Enter a **Name** and select **String** as the **Type**.
   - Select the desired options:
     - **Prompt It?**—Select this option to allow the DTP to be used in a step. The DTP is shown in the template steps that prompt users for input when the template is used in a rule, or when a script template is launched in Planning.
     - **Mandatory?**—Select this option to make it mandatory to enter data for the design-time prompt.
     - **Read Only**—Select this option to make it so that users can only read the prompt.
     - **Is Hidden**—Select this option to hide the design-time prompt from the user. In Planning, on the **Business Rules** tab in form design, add the rule with the template or script template to the form. In the **Business Rules Properties** pane, select **Hide Prompt**, and then select **Use Members on Form**.
   - In **DTP Text**, enter the text that you want users to see when they are prompted for input.

If you select **Prompt It?**, then **DTP Text** is required.

- **Optional.** In **Comments**, enter a comment about the DTP.
- **Optional.** In **Default Value**, click  , then select **DTP**, **Member**, or **Function**, and then enter a DTP, member, or function as the default value.

UDA DTP

Defines a user-defined attribute to use in the design-time prompt.

For example, in Planning, "Account" types are created as UDAs and used this in the Currency Conversion template. *Expense* and *Revenue* accounts are converted with *Average Rates* , and *Assets*, *Liability*, and *Equity* accounts are usually converted with *Ending Rates*.

To create a UDA design-time prompt:

1. In the **Template Designer**, open an existing template or create a new template.
2. On the **Design Time Prompt** tab, click  .
3. If you already have design-time prompts defined for the template, select where to enter the new design-time prompt.
4. In the **Create Prompt** dialog box, on the **Properties** tab, do the following:
   - Enter a **Name** and select **UDA** as the **Type**.
   - Select the desired options:
     - **Prompt It?**—Select this option to allow the DTP to be used in a step. The DTP is shown in the template steps that prompt users for input when the template is used in a rule, or when a script template is launched in Planning.
     - **Mandatory?**—Select this option to make it mandatory to enter data for the design-time prompt.
     - **Read Only**—Select this option to make it so that users can only read the prompt.
     - **Is Hidden**—Select this option to hide the design-time prompt from the user. In Planning, on the **Business Rules** tab in form design, add the rule with the template or script template to the form. In the **Business Rules Properties** pane, select **Hide Prompt**, and then select **Use Members on Form**.
   - In **DTP Text**, enter the text that you want users to see when they are prompted for input.

     If you select **Prompt It?**, then **DTP Text** is required.
   - **Optional.** In **Comments**, enter a comment about the DTP.
   - From the **Dimension** drop down, select a dimension.

     The dimensions that are displayed are the dimensions in the Planning application.
   - **Optional.** In **Default Value**, click the drop down, and the select a default value based on the dimension you selected.
5. On the **Dependencies** tab, define the dependencies. See Defining Dependencies for Design-Time Prompts.
6. On the **Define Limits** tab, select a **Density** and a **Dimension Type**. See Defining Limits for Design-time Prompts

Defining Dependencies for Design-Time Prompts

You can define inclusive and exclusive dependencies for *cross dimension*, *dimensions*, *dimension*, *member*, *members*, and *member range* design-time prompts. If you designate a prompt (for example DTP1) as inclusive of another prompt (for example DTP2), when a user is prompted for DTP1, only the dimensions from DTP2 will display. If you designate a prompt (for example DTP3) as exclusive of another prompt (for example DTP4), when a user is prompted for DTP3, only the dimensions that are not used in DTP4 will display.

> **Note**
>
> Design-time prompts can only be inclusive or exclusive of the design-time prompts that come before them in the design-time prompt grid, so the order of the design-time prompts is very important.

To define dependencies:

1. In **System View**, create or open a custom template.
2. On the **Design Time Prompt** tab, click  , and select where to insert the row.
3. In the **Create Prompt** dialog box on the **Properties** tab, select one of these options in **Type** field:
   - **Cross Dimension**
   - **Dimension**
   - **Dimensions**
   - **Member**
   - **Members**
   - **Member Range**
   - **UDA**

   When you select one of these options, a **Dependencies** tab is displayed.
4. On the **Dependencies** tab, select a design-time prompt, and then select a dependency option:
   - **Exclusive**—Makes the design-time prompt (for example, DTP2) exclusive of another design-time prompt (for example DTP1). When you are prompted for DTP2, only the dimensions that are not used in DTP1 are displayed.
   - **Inclusive**—Makes the design-time prompt (for example, DTP4) inclusive of another design-time prompt (for example, DTP3). When you are prompted for DTP4, only the dimensions that are used in DTP3 are displayed.
   - **None**—No dependencies are defined for the design-time prompt.
5. Click **OK**, and then  **(Save** button).

Defining Limits for Design-time Prompts

When you create a *cross dimension*, *dimension*, *dimensions*, *member*, *members*, or *member range* design-time prompt for a custom template, you must specify whether you want the prompt to display for dense and sparse dimensions, dense dimensions only, or sparse dimensions only.

To define limits:

1. In **System View**, create or open a custom-defined template.
2. On the **Design Time Prompt** tab, click  , and select where to insert the row.
3. In the **Create Prompt** dialog box on the **Properties** tab, select one of these options in **Type** field:
   - **Cross Dimension**
   - **Dimension**
   - **Dimensions**
   - **Member**
   - **Members**
   - **Member Range**
   - **UDA**

   When you select one of these options, a **Define Limits** tab is displayed.
4. On the **Define Limits** tab:
   - In **Density**, select an option:
     - **Show Sparse**—Displays only sparse dimensions for the DTP
     - **Show Dense**—Displays only dense dimensions for the DTP
     - **Both**—Displays both dense and sparse dimensions for the DTP
   - In **Dimension Type** select a dimension type to restrict the design-time prompt to show only the selected dimension types.

     For example, if you only want users to enter a member for *Account*, *Entity*, and *Year*, then select these dimensions in **Dimension Type**. When users are prompted for input, the only dimensions that will show are those with type *Account*, *Entity*, and *Year*.
5. Click **OK**, and then  **(Save** button).

Creating Steps for Design-Time Prompts

Use the Template Wizard Designer to create steps for the template. The wizard enables you to decide what design-time prompts to display in each step, whether to display or hide a step (enabling conditions), and whether to display error or warning messages.(validation conditions).

To create steps for design–time prompts:

1. On the **Design Time Prompt** tab, click  .
2. In the **Template Wizard Designer**, click  to create a step in the wizard.

3. In the **Add Step** dialog box, enter information about the step, and then click **OK**.

   The step you add is displayed in the **Step** drop-down list in the **Template Wizard Designer**.

4. Move the design-time prompts to display in the step from **Available DTPs** to **Selected DTPs**.

5. In the **Enabling Conditions** tab, enter information about whether to display or hide the step:

   - In **Condition Prefix**, select a prefix from the drop-down.
   - In **DTP**, click  to select a DTP or function.
   - In **Operator**, select an operator from the drop-down.
   - In **Value**, enter or select a value for the condition.

   Repeat these steps until you define all of the statements in the condition. To add rows, click the plus icon (+) in last row.

   The first row in the condition is the IF statement; each additional row is an AND statement. For example,

   Each row defaults to an AND statement, but if you click on an AND, it can be changed to an OR.

6. In the **Validation Conditions** tab, enter information about whether to display error or warning messages when users enter data that is not desired (error) or that might not be desired (warning):

   - In **Validation Condition**, click  to define the validation condition.
   - In **Message Level**, select **Error** or **Warning**.

     An error message prevents the next steps. A warning message allows the next step after you click **OK** in the warning message.
   - In **Validation Message**, enter the message that will be displayed to the user in the step.

   Click the plus icon (+) to add additional errors or warnings to the step.

7. Click **OK**, and then  .

Finding and Replacing Text in Design-Time Prompts

You can search for and replace text strings in the design-time prompts of custom templates.

> **⚠ Caution**
>
> The order and location in which you replace design-time prompts is very import. If you want to find and replace a design-time prompt name in both the Design Time Prompt tab and the Template Designer tab, you must first search in the Design Time Prompt tab, save the template, and then find and replace in the Template Designer tab. If you do not perform the search in this order, the design-time prompt name is not replaced in the Template Designer tab.

To search for and replace text in design-time prompts:

1. In **System View**, open a custom template.
2. On the **Design Time Prompt** tab, click  , enter the information to find and replace, and then click **Replace** or **Replace All**.

Opening a Custom Template

You can open a custom template from System View, Deployment View, and Custom View.

You can also open a custom template from a business rule's flow chart in the Rule Designer.

To open a custom template, double-click the template, or right-click the template, and then select **Open.**

Refreshing Custom Templates

After you create a custom template, you may need to refresh the application list in System View to see the new template in the Templates node.

When you refresh the application type, application, or calculation or plan type to which a custom template belongs, you refresh the **Templates** node by default. Refreshing the **Templates** node, however, does not refresh higher levels (that is, calculation or plan types, applications, or application types) in the application list.

To refresh the list of custom templates, right-click the **Templates** node, and then select **Refresh**.

Showing the Usages of a Custom Template

You can display a list of the business rules that are using a custom template.

Viewing the usages of a custom template is useful when you want to delete the custom template and need to know what objects, if any, are using it. You must remove the template from any objects that are using it before you can delete the template.

To show the usages of a custom template, in **System View**, expand the **Templates** node, then right-click the template, and then select **Show Usages**.

Copying and Pasting a Custom Template

You can copy a custom template and paste it into another business rule.

The rule into which you copy it must belong to the same plan type. For example, you can copy a custom template from a rule in a Plan1 plan type in a Planning application to another rule in a Plan1 plan type in a Planning application, but not to a rule in a Capital Asset plan type.

To copy and paste a custom template, do one of the following:

- Expand the **Templates** node, then right-click the template, then select **Copy to**, then enter the requested information in the **Copy To** dialog box, and then click **OK**.
- In the flow chart of a business rule, right-click the template to copy, and then select **Copy**. Open the business rule where you want to paste the template, right-click in the flow chart of the business rule, and then select **Paste**.

Deleting a Custom Template

You can delete a custom template from System View, Custom View, and Deployment View.

Before you delete a custom template, ensure that it is not used in any business rules or any custom folders. To show the usage of a template, right-click the template, and then select **Show Usages**.

To delete a custom template, expand the **Templates** node, then right-click the template, and then select **Delete.**

> **Note**
>
> If a script template has been deployed to Planning, when you select **Delete**, the following message is displayed:
>
> ```
> Do you want to delete the selected item(s)? Some of the selected objects
> were deployed to Planning. Do you want them to be deleted from Planning
> Server?
> ```
>
> Checking the box will delete the script template from both Planning and Calculation Manager.

Finding and Replacing Text in Graphical Custom Templates

You can search for text strings in a custom template. You can also replace all instances of a text string.

When you search for a string, Calculation Manager starts the search with the first component after the component selected in the template's flow chart, continues to the end of the flow chart, and then starts at the beginning of the flow chart until it reaches the component selected in the flow chart. After one occurrence of the text string is found, if you want to search for another, you must start the search again to find the next occurrence.

To search for text or to search for and replace text in a custom template:

1. In **System View**, open a custom template.
2. Do one of the following:
   - On the **Template Designer** tab, select **Edit**, and then **Find**. In the **Find** dialog box, enter the text to find, and then click **Find**.
   - On the **Template Designer** tab, select **Edit**, and then **Replace All**. In the **Replace** dialog box, enter the information to find and replace, and the select **Replace All.**

7

Using Components to Design Business Rules and Templates

Related Topics

- About Components
  Business rules and templates can include several types of components.
- Formula Components
  A Formula component is comprised of formula calculation statements.
- Script Components
  Script components can be used in business rules and templates.
- Condition Components
  A condition component is comprised of conditional statements that are either true or false.
- Member Block Components
  A member block component defines the member to surround one or more statements in a script.
- Member Range Components
  A member range component is a type of loop comprised of a range of members from Planning dimensions.
- Fixed Loop Components
  A fixed loop component is an object that cycles through a list of metadata members a fixed number of times.
- Metadata Loop Components
  Metadata loop components enable you to assign a value to multiple members using either a function (metadata) or a start and end value (fixed).
- DTP Assignment Components
  Use a DTP Assignment Component to assign a DTP, member, function, or typed text to a design-time prompt in a custom template.
- Sharing Script and Formula Components
  A shared formula or script component exists at the formula level and script level.
- Copying Components
  Copy and paste the children of a business rule component, or copy and paste the reference to a business rule formula or script component.
- Saving Components
  You save formula and script components after you design them in the Component Designer.
- Refreshing Formula and Script Components
  After you create a formula or script component, you may need to refresh the application list in the System View to see it in the Formulas or Scripts node.
- Showing the Usages of Formula and Script Components
  You can show the usages of script or formula components.
- Working with Components in a Flow Chart
  You can perform actions on the components in a flow chart.

About Components

Business rules and templates can include several types of components.

- **Formula Components**—Calculation statements that you can write or design using members, functions, and conditional statements
- **Script Components**—Only calc script statements.
- **Condition Components**—Conditional statements (that is, If...Then statements) that are either true or false.
- **Member Block Components**—Contain one member that you specify.
- **Member Range Components (or metadata loops)**—Contain lists of metadata members (for example, lists of accounts).
- **Fixed Loop Components**—Contain loops of metadata that, for example, loop through a list of members like accounts.
- **Metadata Loop Components**—(custom templates users only) Contain a parameter or parameters that you can apply to the children of a parent dimension.
- **DTP Assignment Components**— (custom templates users only) Contain design-time prompts and the conditions that you define for them.

You can create formula and script components independently of the rules and templates in which they are used. Because they are independent objects, you can open, save, edit, delete, and export them from within the System View.

Unlike formula and script components, you must create the other component types from within rules and templates. You cannot open, save, delete, or export them independently of the rules and templates to which they belong.

> **Note**
>
> As you create components, you may want to leave the business rules, components, templates and variables you are working with open. Calculation Manager displays these objects in a tabbed interface so you can move easily among the tabs as you are creating components. You can have as many as ten tabs open within Calculation Manager, but Oracle recommends that you not open more than ten objects simultaneously for optimum performance.

Formula Components

A Formula component is comprised of formula calculation statements.

Related Topics

- About Formula Components
- Creating a Formula Component
- Designing a Formula Component
- Opening a Formula Component
- Editing a Formula Component

About Formula Components

A formula component is comprised of formula calculation statements. To create the calculation statements of a formula, you enter or select members, functions, and conditions. As you create the formula, each of its calculation statements is listed in a row within a grid in the Component Designer.

Creating a Formula Component

You can create a formula component from the System View. Formula components can be used in business rules and templates.

To create a formula component:

1. In **System View**, click the **New Object** icon.
2. Enter the **Application Type**.
3. Select an **Application**. The application must be a valid Planning application.
4. Select the **Plan Type**.

> **Note**
>
> From the System View, if you right-click Formulas and select New to create a new formula, the New Formula dialog is populated with the application type name, the application name, and the plan type name in which you are working.

5. Select **Formula** as the **Object Type**.
6. Enter the formula's name, and then click **OK**.

   The formula is displayed in the Component Designer. To design the formula component, see Designing a Formula Component.

Designing a Formula Component

You can create a formula component from the System View or any other view. You can also create a formula component from within the Rule or Template Designer as you are designing a business rule or template. Formula components can be used in business rules and templates.

To create a formula component:

1. In **System View**, right-click **Formulas**, and then select **New**.

   The **New Formula** dialog is populated with the Planning application type, the Planning application, and the plan type.

> **Note**
>
> You can also create a formula component from within a business rule or template by dragging a new formula component into the business rule or template flow chart.

2. Enter a name, application type, application, and plan type to which the formula component belongs.
3. Click **OK**.
4. **Optional:** On the **Component Designer**, you can perform any of these tasks:
   - Click **Add Grid**, and select **Insert Grid Before**, **Insert Grid After**, or **Insert Grid At End** to add another grid. By default, the Formula tab displays one grid.
   - Click **Delete Grid** to delete a selected grid.
   - Click the **Find** icon to find text in the formula grid in which you are working. Perform these tasks:
   - Click the **Find and Replace** icon to find and replace text within the script.
5. On **Formula**, enter a caption for the formula.
6. **Users creating a formula component for a template only:** On the **Formula** tab, select **Use Design Prompt** if you want to use a design-time prompt in the formula component. Then in the condition grid, define a condition for the design-time prompt by performing these tasks:
   a. In **DTP**, select a design-time prompt.
   b. In **Operator**, select an operator: = = or <>.
   c. In **Value**, select a value from the dropdown list.
   d. Repeat these steps to create as many statements in the condition as you need.

   > **Tip**
   >
   > Click the plus (+) and minus (-) icons to add and delete rows from the grid. Change the And to Or by clicking in the field next to And; select Or from the dropdown list.
7. By default, processing of a formula component calculation starts with the first member you enter in the grid. If you want to start processing the formula component calculation with another member, enter the member or function name in **Member Block**, or click the **Ellipsis** icon to pick one from the Member Selector. See About Adding Members and Functions to a Component.
8. In **Comment**, enter comments for the conditional and formula statements you want to create.
9. **Optional:** To create a conditional statement (that is, an IF statement) for the formula component, click **Add Condition**. See Using the Condition Builder to Create Conditional Statements.
10. **Optional:** In the IF row that is displayed, enter the text of the condition statement, or click the **Add/Edit** condition icon in the right column of the row to access the Condition Builder. The Condition Builder enables you to design a condition statement graphically.

> **Note**
>
> Though you can select IF, ELSE IF, and ELSE from the down arrow, by default, the first statement must be an IF statement.

11. **Optional:** Use the Condition Builder to design the IF conditional statement. See Using the Condition Builder to Create Conditional Statements.
12. In the **Formula** row, click in the row to the *left* of the equal sign to create a formula statement. Click the **Actions** icon, and select:

> **Note**
>
> If you created a conditional statement (that is, an IF statement) in step 9 through step 11, the formula statement you create in this step is the THEN statement of the condition.

    - Variable (See Working with Variables.)
    - Member (See Adding Members and Functions to a Component.)
    - Function (See Working with Functions.)
    - Smart List (See Working with Smart Lists.)
13. In the **Formula** row, click in the row to the *right* of the equal sign to complete the formula statement. Click the **Actions** icon, and select:
    - Variable (See Working with Variables.)
    - Member (See Adding Members and Functions to a Component.)
    - Function (See Working with Functions.)
    - Smart List (See Working with Smart Lists.)
14. Click the **Comments** icon to enter comments about the calculation statement row. Click **OK**.
15. **Optional:** If you want to create another IF statement, or an ELSE IF or ELSE statement, complete these steps:
    a. Click **Add Condition**.
    b. Click the **down arrow**, and select one of these options:
        - **IF**: select this to create an IF conditional statement. If the IF statement is TRUE, then actions are performed; if the IF statement is FALSE, then other actions are performed.
        - **ELSE IF**: select this to create an ELSE IF conditional statement. The actions in this statement are performed if there is an associated IF statement and the IF statement is FALSE.
        - **ELSE**: select this to create an ELSE conditional statement. The actions in this statement are performed if there is an associated ELSE IF statement and the ELSE IF statement is FALSE.

> **Note**
>
> If you remove the condition statement from an IF or ELSE IF block, and if the next block contains an ELSE IF statement, then the next block is changed to an IF block. However, if the next block contains an ELSE statement, then the condition statement in this block is removed.

c. Repeat step 9 through step 14 to design additional statements for the formula component. A formula grid can consist of one or more formula blocks that contain a collection of formula statements. You can also enter a comment and a condition for the block, though this is optional.

> **Tip**
>
> If necessary, click the **+** icon to add more formula rows.

16. On **Properties**, complete these steps:

    a. **Users creating a formula component for a business rule only:** When you drag and drop the formula component into a business rule from **New Objects**, **Shared** is not selected. To make this formula shared, select the **Properties** tab of the formula, select **Shared**, and name the formula.

    When you create a formula component from the **System View**, **Shared** is selected by default, and you cannot change it. If, instead, you want a copy of the formula in a business rule, drag the shared formula into the business rule, then clear the **Shared** check box on the **Properties** tab. This creates a copy of the shared formula in the business rule; the copy of the shared formula is not changed if the shared formula is changed. When you save the business rule, the formula no longer has a name.

    b. **Optional:** Edit the formula's name by entering a new one of up to 50 characters. (The name defaults from the New Formula dialog.)

    c. **Optional:** Enter a description of up to 255 characters for the formula.

    d. **Optional:** Enter a caption for the formula. The caption is displayed below the formula in the **Designer** and **Template Designer** flow charts.

> **Note**
>
> If the component does not have a caption, and the component is not shared, the first part of the component is shown in the flow chart. For example, if you have a formula, "Salaries" = 40, that is not shared and does not have a caption, then the flow chart shows "Salaries" = 40 for the formula component. If the component is shared, and does not have a caption, the name of the component is shown in the flow chart.

    e. Enter comments for the formula. For example, you may want to tell users what the formula should be used for.

17. **Users creating a formula component for a business rule only:** On **Usages**, you can view the rules that use the formula component.

> **Note**
>
> None of the information on the Usages tab can be edited.

18. Click  .

Using the Condition Builder to Create Conditional Statements

The Condition Builder enables you to create conditional statements for formula and condition components. Conditional statements are also known as If...Then statements. If the first part (the If part) of a conditional statement is true, then the second part of the statement (the Then part) is also true. If the first part of a conditional statement is false, then the second part may or may not be true.

To create a conditional statement:

1. Right-click a formula, and then select **Open**.
2. In the **Component Designer**, click the **Add Condition** icon.

   Two rows with various fields and drop-down lists that you use to build the condition are added to the Component Designer.
3. In the first row, select whether the first statement in the condition should begin with IF, ELSE, or ELSE IF, and enter the text of the conditional statement. Or click the **Add/Edit Condition** icon to the right of the row to access the Condition Builder. The Condition Builder enables you to design a condition statement graphically.

   Although you can select IF, ELSE IF, and ELSE from the down arrow, by default, the first statement must be an IF statement.
4. In the Condition Builder, select **Metadata Condition** or **Data Condition**.
5. Do one of these tasks:
   - Enter a value in **Function** (for a metadata condition) or **Formula** (for a data condition).
   - Click in the **Function** or **Formula** row to display the **Actions** icon. Click the **Actions** icon, and select an option to create a formula or function:
     - Variable (See Working with Variables.)
     - Member (See Adding Members and Functions to a Component.)
     - Function (See Working with Functions.)
6. Enter one of these operators:
   - = (equal to)
   - < (greater than)
   - > (less than)
   - <> (greater than or less than)
   - >= (less than or equal to)
   - <= (greater than or equal to)
7. Do one of these tasks:
   - Enter a value in **Value**.

- Click in the **Value** row to display the **Actions** icon. Click **Actions**, and select an option to create a value for the formula or function:
  - Variable (See Working with Variables.)
  - Member (See Adding Members and Functions to a Component.)
  - Function (See Working with Functions.)

8. Enter any comments for the condition.
9. Click the **+** icon to add the condition to the Condition grid. (You can also use the **+** icon to create a copy of a selected row, and add it to the Condition grid.) Use the **—** icon to replace a selected row in the Condition grid with a metadata or data condition.

> **Tip**
>
> Use the up arrow and down arrow icons to reorder the condition statements in the grid. Use the Group and Ungroup icons to group (add parentheses to) and ungroup (remove parentheses from) parts of the condition statement. Change the And to Or, And Not, or Or Not by clicking in the field next to And, and using the drop down to select an option.

10. Click **OK** to exit the **Condition Builder** dialog.

    The condition is inserted into the IF statement.
11. **Optional:** Repeat step 2 through step 10 for each condition statement you want to create.
12. **Optional:** For each additional condition statement, double-click in the first column to display a dropdown list from which you can select an operator to start each condition statement:
    - IF: You can start *only* the first condition statement with IF. (This is the default that displays only for the first condition statement you create.)
    - NOT IF: You can start *only* the first condition statement with NOT IF.
    - AND: You can start any condition statement (except the first) with AND when you want to create a compound of at least two condition statements.
    - OR: You can start any condition statement (except the first) with OR when you want to create a compound of at least two condition statements.
    - AND NOT: You can start any condition statement (except the first) with AND NOT when you want to include the condition statement that follows it with the formula component.
    - OR NOT: You can start any condition statement (except the first) with OR NOT when you want to exclude the condition statement that follows it from the formula component.

> **Tip**
>
> Use the Group and Ungroup icons to add and remove parentheses from condition statements. To group or ungroup multiple condition statements, use Ctrl + Click or Shift + Click to select the condition statements you want to group or ungroup.

13. Click **OK**.

    The condition statements are inserted into the Condition row.

Entering Comments for Formula Statements

To enter comments for formula statements:

1. With **Formulas** expanded, do one of these tasks:
   - If the formula for which you want to add comments is in a business rule, expand **Rules**, right-click the rule containing the formula component, and select **Open**.
   - If the formula for which you want to add comments is in a template, expand the **Templates** node, right-click the rule containing the formula component, and select **Open**.
   - If you want to open the formula by itself, expand the **Formulas** node, right-click the formula, and select **Open**.
2. Do one of these tasks:
   - If you are adding comments for a formula component in a business rule, in the **Rule Designer** flow chart, select the formula component and the **Formula** tab. Then enter comments in the **Comment** text box.
   - If you are adding comments for a formula in a template, in the **Template Designer** flow chart, select the formula component and the **Properties** tab. Then enter comments in the **Comments** text box.
   - If you are adding comments for a formula component by itself, in the **Component Designer**, enter comments in the **Comments** text box.
3. Click  .

Opening a Formula Component

You can open a formula component from the System View or from within the Rule Designer or Template Designer flow chart.

To open a formula component, in **System View**, expand **Formulas**, then right-click a formula, and then select **Open**.

> **Note**
>
> If the formula component is used in a business rule, you can also open it from within the rule's flow chart in the Rule Designer by right-clicking the formula component and selecting Open, or by double-clicking it.

Editing a Formula Component

You can edit the statements that comprise a formula component and the formula component's comments, caption, name, and description.

To edit a formula component:

1. In **System View**, right-click a formula, and then select **Open**.
2. In **Component Designer**, edit the formula properties, and then click  .

See Designing a Formula Component and Using the Condition Builder to Create Conditional Statements.

Deleting a Formula Component

You can delete a formula component only if it is not being used in any rules or templates. To see if any rules or templates are using the formula component, you can show its usages. (See Showing the Usages of Formula and Script Components.) If a formula component is used in a rule or template, and you no longer need to use it in that rule or template, you can remove it from the rule or template, and then delete the formula component. You can also delete the rule or template, which deletes the formula component within it.

To delete a formula component:

1. In **System View**, right-click a formula, and then select **Show Usages**.

   If any are, you must remove the formula component from them.
2. Ensure no rules or templates are using the formula component.

   See Removing a Component from a Flow Chart.
3. Right-click the formula again, and then select **Delete**.

Copying and Pasting a Formula Component

You can copy a formula component from a rule or template and paste it into the same, or a different, rule or template. You can also copy the contents of the condition grid within a formula component and paste it into the same, or a different, formula component. You cannot copy a formula component and paste it into another formula component or another component type.

To copy and paste a formula component:

1. In **System View**, right-click a rule or template, and then select **Open**.
2. In the Rule Designer or Template Designer flow chart, right-click a formula component, and then select **Copy**.

   If the component you want to copy is shared, you can right-click the formula component and select Copy Reference to copy the reference to the shared component instead of copying the component itself. (See Copying and Pasting the Reference to a Business Rule Formula or Script Component.)

   If the component you want to copy contains other components (that is, children), you can copy all of the components in the group by using Ctrl+click to select them all, right-clicking them, and selecting Copy Group. (See Copying and Pasting a Component Group in a Flow Chart.)
3. Do one of these tasks:
   - To paste the formula component into the *same* business rule or template flow chart, right-click in the location of the flow chart into which you want to paste the formula component, and select **Paste**.)
   - To paste the formula component into a *different* business rule or template flow chart, open the business rule or template into which you want to paste the component, right-click in the location of the flow chart into which you want to paste the formula component, and select **Paste**.
4. Click  .

Script Components

Script components can be used in business rules and templates.

Related Topics

- Creating a Script Component
- Designing a Script Component
- Opening a Script Component
- Editing a Script Component
- Deleting a Script Component
- Copying and Pasting a Script Component

Creating a Script Component

You can create a script component from the System View. Script components can be used in business rules and templates.

To create a script component:

1. Enter the script's name and application type.
2. Select an **Application Name**. The application name must be a valid Planning application.
3. Select the **Plan Type**, and then click **OK**.

> **Note**
>
> From the System View, if you right-click Scripts, and select New, the New Script dialog is populated with the application type, the application, and the plan type with which you are working.

Designing a Script Component

Script components can be used in business rules and templates. You create a script component from the System View or any of the other views. You can also create a script component from within the Rule or Template Designer while you are designing a business rule or template.

> **Note**
>
> You can also design a script component from within a business rule or template.

To design a script component:

1. In **System View**, right-click **Scripts**, and then select **New**.
2. Enter a name, application type, application, and plan type for the script.
3. Click **OK**.

4. Use the icons to design the script component.

> **Note**
>
> Unlike rules, templates, and other components, when you open a script, you cannot view it in a graphical format (within a flow chart). You can view it only in script format.

You can do any of these tasks with the script:

- Click the **Hide/Show Line Numbers** icon to hide (or display) the script line numbers. Line numbers are displayed by default.
- Click the **Insert a function and its parameters** icon to insert a function into the script. See Working with Functions.
- Click the **Insert members selected from a dimension** icon to insert a member into the script. See Adding Members and Functions to a Component.
- Click the **Insert a variable** icon to insert a variable into the script. See Working with Variables.
- Click **Insert smartlists** to insert a Smart List into the script. See Working with Smart Lists.
- Click the **Comment** and **Uncomment** icons to add and remove comment lines from the script.
- Click the **Hide Comments** icon to hide the system-generated comments in the script.
- Click the **Verify Syntax** icon to check the syntax of the script for errors.

> **Note**
>
> If there is a syntax error, the code containing the error changes to red text, there are no error messages displayed. If there is not a syntax error, the message, "No syntax error(s) found," is displayed.

- Click the **Wrap** icon so that any long lines of the script that scroll off the page display on multiple lines on the same page.
- Click the **Set Code Completion Off** icon to suppress suggestions for completing the code. (This icon functions as a toggle. To turn code completion back on, click the icon again. By default, code completion is set to On.)
- Click the **Replace** icon to *find and replace* a text string in the script. See Searching for a Text String in a Business Rule Script.
- Click the **Find** icon to *find* a text string in the script. See Searching for a Text String in a Business Rule Script.
- Enter search text in **Find**, and click **Previous** or **Next** to find the previous (by searching up in the script) or next (by searching down in the script) instance of the text.
- Enter a line number in **Go to Line**, and click the arrow to highlight the line number in the script.
- Expand or collapse a line in the script by clicking the plus or minus sign to the left of the line. For example, FIX statements display with all of the lines in the statement

displayed by default, but you can hide the lines of the FIX statement by clicking the minus sign to the left of it.

- Select the lines of the script to copy, and click the **Copy to Clipboard** icon.
- Select the lines of the script to cut, and click the **Cut to Clipboard** icon.
- Place the cursor in the location of the script where you want to paste, and click the **Paste from Clipboard** icon.
- Auto suggest is provided for functions. When you type the first few characters of the function, press **CTRL+Space** to display the suggestions. Select one of the suggestions, and click or press **Enter** to complete the function.

5. On **Properties**, complete these steps:
   a. When you create a script component, **Shared** is not selected. To make the script component shared, select the **Properties** tab, select **Shared**, and type in a name.

      When you create a script component from the System View, **Shared** is selected by default, and you cannot change it. If, instead, you want a copy of the script in a business rule, drag the shared script into the business rule, then clear the **Shared** check box on the **Properties** tab. This creates a copy of the shared script in the business rule; the copy of the shared script is not changed if the shared script is changed. When you save the business rule, the script no longer has a name.

   b. **Optional:** Edit the script's name by entering a new one of up to 50 characters. (The name defaults from the New Script dialog.)

      > **Note**
      >
      > You can only name a shared script. If the script is not shared, you cannot type in the Name field.

   c. Enter a description of up to 255 characters for the script.
   d. Enter a caption for the script. The caption is displayed below the script in the **Rule Designer** and **Template Designer** flow charts.
   e. Enter comments for the script. For example, you may want to tell users what the script should be used for.
6. On **Script**, enter a caption for the script component.
7. **Users creating a script component for a template only:** Select **Use Design Prompt** if you want to use a design-time prompt in the script component. Then in the condition grid, define a condition for the design-time prompt by performing these tasks:
   a. In **DTP**, select a design-time prompt.
   b. In **Operator**, select an operator: = = or <>.
   c. In **Value**, use the dropdown list to select a value. You cannot type in this field.
   d. Repeat these steps to create as many statements in the condition as you need.

      > **Tip**
      >
      > Click the plus (+) and minus (-) icons to add and delete rows from the grid.

8. **Users creating an independent script component (that is, a script component that is not within a business rule or a template) or a script component for a business rule only:** On **Usages**, you can view the rules that use the script component.

> **Note**
>
> You cannot edit any of the information on this tab.

9. Click  .

Opening a Script Component

You can open a script component from the System View, or in a flow chart in the Rule Designer or Template Designer.

To open a script component:

1. In **System View**, expand **Scripts**.
2. Right-click the script, and then select **Open**.

> **Note**
>
> If a shared script component is used in a business rule, you can also open it from within the rule's flow chart in the Rule Designer by right-clicking the script component and selecting Open, or by double-clicking it.

Editing a Script Component

You can edit the functions, variables, and members you use to write the script component and the script component's comments, caption, name, and description.

To edit a script component:

1. In **System View**, right-click a script, and then select **Open**.

> **Note**
>
> Unlike rules, templates, and other components, when you open a script, you cannot view it in a graphical format (within a flow chart). You can view it only in script format.

2. Edit the desired script properties and then click  .

   See Designing a Script Component.

Deleting a Script Component

You can delete a script component only if it is not being used in any rules or templates. To see if any rules or templates are using the script component, you can show its usages. (See

Showing the Usages of Formula and Script Components.) If a script component is used in a rule or template, and you no longer need to use it in that rule or template, you can remove it from the rule or template, and then delete the script component. You can also delete the rule or template, which deletes the script component within it.

To delete a script component:

1. In **System View**, right-click the script, and then select **Show Usages**.
2. Ensure that no rules or templates are using the script component

   If any are, you must remove the script component from them. See Removing a Component from a Flow Chart.
3. Right-click the script again, and then select **Delete**.

Copying and Pasting a Script Component

You can copy a script component from a rule or template and paste it into the same, or a different, rule or template. You can also copy *the script within a script component* and paste it into the same, or a different, script component. You cannot copy a script component and paste it into another script component or another component type.

To copy and paste a script component::

1. In **System View**, select the rule or template that contains the script component.
2. In the **Rule Designer** or **Template Designer** flow chart, right-click the script component, and then select **Copy**.

   > **Note**
   >
   > If the component you want to copy is shared, you can right-click the script component, and select Copy Reference to copy the reference of the shared component instead of copying the component itself. (See Copying and Pasting the Reference to a Business Rule Formula or Script Component.)

3. Do one of these tasks:
   - To paste the script component into the *same* business rule or template, right-click the location in the flow chart where you want to paste it, and select **Paste**.
   - To paste the script component into a *different* business rule or template, open the business rule or template into which you want to paste the script component, right-click the location in the flow chart where you want to paste it, and select **Paste**.
4. Click  .

Condition Components

A condition component is comprised of conditional statements that are either true or false.

Related Topics

- About Condition Components
- Creating a Condition Component
- Opening a Condition Component

- Editing a Condition Component
- Deleting a Condition Component
- Copying and Pasting a Condition Component

About Condition Components

A condition component is comprised of conditional statements (that is, IF...THEN statements) that are either true or false. If the condition is true, the system performs the actions you specify; if the condition is false, the system performs other actions you specify. The condition can be a metadata condition or a data condition. Condition components cannot be shared.

Creating a Condition Component

You create condition components from within business rules or templates. Unlike script and formula components, condition components cannot be created as independent objects. They are linked to the business rule or template for which they are created. They cannot be shared.

To create a condition component:

1. In **System View**, right-click a rule or a template, and then select **Open**.
2. After you determine where in the flow chart you want to create the condition, from the **New Objects Palette**, drag the **Condition** component and drop it into that location in the flow chart.

   The Condition object is displayed as a diamond with a question mark in the flow chart.
3. **Optional:** On **Condition**, enter a caption to identify the condition component. The caption is displayed above the component in the flow chart of any rule or template that uses the condition.
4. **Users creating a condition component for a template only:** Select **Use Design Prompt** if you want to use a design-time prompt in the condition component. Then in the condition grid, define a condition for the design-time prompt by performing these tasks:
   a. In **DTP**, select a design-time prompt.
   b. In **Operator**, select an operator: = = or <>.
   c. In **Value**, use the dropdown list to select a value. You cannot type in this field.
   d. Repeat these steps to create as many statements in the condition as you need.

   > **Note**
   >
   > If you use a design-time prompt to define the condition, you cannot enter a condition in, or use the Condition Builder to build a condition for, the Condition box.
5. If you want to create a member block, click the **Ellipsis** icon, and select a member or function. By default, processing of a condition component calculation starts with the first member you enter in the grid. If you want to start processing with another member, enter the member or function name in **Member Block**, or click the **Ellipsis** icon to pick one from the Member Selector. See About Adding Members and Functions to a Component
6. Do one of these tasks:
   - Enter the condition statements in the Condition text box.

- Click **Add Condition** to use the Condition Builder to create the condition statements. See Using the Condition Builder to Create Conditional Statements.

7. Enter comments for the condition component, and then click  .

Opening a Condition Component

You open a condition component from within the flow chart of the business rule or template to which it belongs. Unlike formula and script components, you cannot open it from the System View.

To open a condition component:

1. In **System View**, right-click a rule or a template, and then select **Open**.
2. When the rule or template opens, select the condition component in the flow chart to see the condition properties.

Editing a Condition Component

You can edit what functions, variables, and members you use to create the condition component and the condition component's comments and caption.

To edit a condition component:

1. In **System View**, right-click a rule or a template, and then select Open.
2. In the **Rule Designer** or **Template Designer**, select the condition to edit its properties in **Condition**. You can edit any of these properties. (See Creating a Condition Component.)
   - The caption
   - The condition statements
   - The comments
   - **Templates users only:** The design-time prompts
3. Click  .

Deleting a Condition Component

You delete a condition component by removing it from the business rule or template to which it belongs. Since a condition component can be used in only one business rule or template, you delete it simply by removing it from the business rule or template.

To delete a condition component:

1. In **System Designer**. right-click a rule or a template, and then select **Open**.
2. In the **Rule Designer** or **Template Designer**, select the condition component you want to delete in the flow chart.
3. Right-click the condition, and then select **Remove**.

   Condition components cannot be shared so when you remove a condition component from a business rule or template, it is deleted from the database.
4. Confirm deletion of the component, and then click  .

Copying and Pasting a Condition Component

You can copy a condition component from a rule or template and paste it into the same, or a different, rule or template. You cannot copy a condition component and paste it into another condition component or another component type.

To copy and paste a condition component from the Rule Designer or Template Designer:

1. In **System View**, right-click a rule or a template, and then select **Open**.
2. In the Rule Designer or Template Designer flow chart, right-click the condition component you want to copy, and then select **Copy**.
3. Do one of these tasks:
   - To paste the condition component into the *same* business rule or template, right-click the location in the flow chart where you want to paste the condition component, and select **Paste**.
   - To paste the condition component into a *different* business rule or template, open the business rule or template into which you want to paste the component, right-click the location in the flow chart where you want to paste the condition component, and select **Paste**.
4. Click  .

Member Block Components

A member block component defines the member to surround one or more statements in a script.

Related Topics

- About Member Block Components
- Creating a Member Block Component
- Opening a Member Block Component
- Editing a Member Block Component
- Deleting a Member Block Component
- Copying and Pasting a Member Block Component

About Member Block Components

A member block component defines the member to surround one or more statements in a script. For example, an If condition needs to be surrounded by a member block. Member blocks can be used in both business rules and custom templates.

A member block is required if you are using an Oracle Essbase substitution variable or a cross dimension member in the target of a formula. A member block can also be used if the member is repeated in the consecutive statements of a formula's target.

Member block components do not exist as database objects that are independent of the business rule or template in which they are created; they exist only as part of the template or business rule to which they belong. Member blocks cannot be shared among rules and templates.

Creating a Member Block Component

You create a member block component from within the Rule or Template designer as you are designing a rule or template. You cannot create a member block component from within the System View.

To create a member block component:

1. In **System View**, right-click a rule or a template, and then select **Open**.
2. After you determine where in the flow chart you want to create the member block, from **New Objects**, drag the member block object, and drop it into the flow chart.

   The member block object is displayed as four blocks with a connecting line in the flow chart.
3. **Optional:** If you are creating a member block component for a template, you can also create design-time prompts for it. See Creating Design-Time Prompts for Custom Templates.
4. On the **Member Block** tab, next to **Member Block**, click the **Actions** icon, and do one of the following:
   - Select **Variable** to search for or create a variable, and then click **OK**. See Working with Variables.
   - Select **Member** to search for a member, and then click **OK**. See About Adding Members and Functions to a Component.
5. On the **Errors and Warnings** tab, click the button to run script diagnostics.
6. In **Properties**, select **Disabled** to exclude the member block from validation and calculation. Then enter an optional description and comments.
7. Click  .

Opening a Member Block Component

You open a member block component from within the flow chart of the business rule or template in which it is used. You cannot open it from the System View.

To open a member block component:

1. In **System View**, right-click a rule or a template, and then select **Open**.
2. When the rule or template opens, select the member block component in the flow chart to see the member block properties.

Editing a Member Block Component

You can change the member in the member block and the member block component's design-time prompts (for custom template users only).

To edit a member block component:

1. In **System View**, right-click a rule or a template, and then select **Open**.
2. In the **Rule Designer** or **Template Designer**, select the member block in the flow chart to edit its properties on **Member Block**.

You can change the member or variable that you selected for the member block, the description of and comments about the member block, and whether the member block is disabled so it is not included in validation and calculation of the business rule or template.

3. Click  .

Deleting a Member Block Component

You delete a member block component by removing it from the business rule or template to which it belongs. You can use a member block component in only one business rule or template; it cannot be shared among other business rules or templates.

To delete a member block component:

1. In **System View**, right-click a rule or a template, and then select **Open**.
2. In the **Rule Designer** or **Template Designer**, select the member block component that you want to delete in the flow chart.
3. Right-click the member block, and then select **Remove**.

   A member block only exists in the business rule or template for which you created it, so when you remove a member block, it is deleted from the database.

Copying and Pasting a Member Block Component

You can copy a member block component from a rule or template and paste it into the same, or a different, rule or template. You cannot copy a member block component and paste it into another member block component or another component type.

To copy a member block component from the Rule Designer or Template Designer:

1. In **System View**, right-click a rule or a template, and then select **Open**.
2. In the **Rule Designer** or **Template Designer** flow chart, right-click the member block component that you want to copy, and then select **Copy**.
3. Do one of these tasks:
   - To paste the member block component into the *same* business rule or template, right-click the location in the flow chart where you want to paste it, and select **Paste**.
   - To paste the member block component into a *different* business rule or template, open the business rule or template into which you want to paste the component, right-click the location in the flow chart where you want to paste it, and select **Paste**.
4. Click  .

Member Range Components

A member range component is a type of loop comprised of a range of members from Planning dimensions.

Related Topics

- About Member Range Components
- Creating a Member Range Component
- Opening a Member Range Component

- Editing a Member Range Component
- Deleting a Member Range Component
- Copying and Pasting a Member Range Component

About Member Range Components

A member range component is a type of loop comprised of a range of members from Planning dimensions. Member range components cannot be shared, so you need to create a new member range component each time you add one to a business rule or template.

Creating a Member Range Component

You create a member range component from within the Rule or Template designer as you are designing a rule or template. Unlike script and formula components that exist independently of the business rules and templates they are used in, you cannot create a member range component from the System View. Member range components are linked to the business rules and templates to which they belong; they cannot be shared.

To create a member range component:

1. In **System View**, right-click a rule or a template, and then select **Open**.
2. From **New Objects**, drag a member range object, and drop it into the flow chart.

   The member range object is displayed as two circles with a connecting line.
3. **Custom template users only:** If you are creating a member range component for a template, create design-time prompts for it on the **Design-Time Prompt** tab. See Creating Design-Time Prompts for Custom Templates.
4. On the **Member Range** tab, do one of the following:
   - Select **Variable Selector** to use a variable to define the member range. Then enter or select a variable. See Working with Variables.

     > **Note**
     >
     > If you select this option, and you change a member, you receive a message: "Editing the grid will remove the dynamic variable linking. Are you sure you want to continue?" If you want to remove the link to the variable, click Yes. Clicking Yes removes the link to the variable and leaves the member range with the members that were in the variable. If you do not want to lose the link to the variable, click No.
     >
     > if you select this option, select **Link Variable Dynamically** so whenever changes are made to the variable. the variable in the rule or template is updated with these changes.

   - Perform these steps:
     a. Enter members in the **Value** column for each dimension you want to define a member range for, or click in each row to display the **Actions** icon.

> **Tip**
>
> Rather than selecting members for each dimension, one row at a time, you can click the Member Selector icon to select members for all dimensions in the grid. When you click OK in the Member Selector, the members you selected are displayed for each dimension in the grid for which you selected members.

b. Click the **Actions** icon, and select one of these options to define the member range:
   - Variable (See Working with Variables.)
   - Member (See Adding Members and Functions to a Component.)
   - Function (See Working with Functions.)

c. Click the **Comments** icon to enter comments for the member range.

d. Repeat these steps for each dimension for which you want to define a member range.

e. Select **Exclude Grid Values** to exclude the members you select from calculation.

f. Click **Reset Grid** to clear any members, variables, and functions you entered in the grid.

g. Select **Enable Parallel Processing** to process the member range commands simultaneously, instead of sequentially. (By default, serial processing is used, but parallel processing may optimize the calculations.) Then in the text box, enter the number of threads to be available for parallel processing, or click the **Variable Selector** to select a numeric or integer type variable if you are creating the member range in a business rule, or a numeric design-time prompt, if you are creating the member range for a template.

   By default, the number of threads for parallel processing is two. For 32-bit platforms, select an integer between 1-4. For 64-bit platforms, select an integer between 1-8.

5. **Business rules users only:** On the **Errors and Warnings** tab, click the button to run script diagnostics.

6. **Optional:** On **Properties**, select **Disabled** to exclude the member range (and any components within it) from validation. You may want to use this for troubleshooting when a component in a business rule or template is preventing the business rule or template from validating.

   You can also enter a description, comments, and a caption for the member range. The caption is displayed below the component in the flow chart of the rule or template to which it belongs.

7. Click  .

Opening a Member Range Component

You open a member range component from within the flow chart of the business rule or template in which it is used. Unlike formula and script components, you cannot open it from the System View.

To open a member range component:

1. In **System View**, right-click a rule or a template, and then select **Open**.
2. When the rule or template opens, select the member range component in the flow chart to see the member range properties.

Editing a Member Range Component

You can edit the dimensions and members, or the variables, you use to define the member range component, whether the time taken to process the member range component is recorded in the log file (for Oracle Hyperion Financial Management applications only), and the member range component's comments and caption.

To edit a member range component:

1. In **System View**, right-click a rule or a template, and then select **Open**.
2. In the **Rule Designer** or **Template Designer**, select the member range to edit its properties on **Member Range**. You can edit these properties of a member range. (See Creating a Member Range Component.)
   - Caption, description, and comments
   - Variables you select to define the member range
   - Dimensions you include in the member range
   - Members that define the member range for each dimension
3. Click  .

Deleting a Member Range Component

You delete a member range component by removing it from the business rule or template to which it belongs. A member range component can be used in only one business rule or template, so you delete it simply by removing it from the business rule or template.

To delete a member range component:

1. In **System View**, right-click a rule or a template, and then select **Open**.
2. In the **Rule Designer** or **Template Designer**, select the member range component you want to delete in the flow chart.
3. Right-click the member range, and then select **Remove**.

   Removing the member range deletes it from the database.

Copying and Pasting a Member Range Component

You can copy a member range component from a rule or template and paste it into the same, or a different, rule or template. You cannot copy a member range component and paste it into another member range component or another component type.

To copy a member range component from the Rule Designer or Template Designer:

1. In **System View**, right-click a rule or a template, and then select **Open**.
2. In the **Rule Designer** or **Template Designer**, right-click the member range component you want to copy in the flow chart, and then select **Copy**.
3. Do one of these tasks:

- To paste the member range component into the *same* business rule or template, right-click the location in the flow chart where you want to paste it, and sthen elect **Paste**.
- To paste the member range component into a *different* business rule or template, open the business rule or template into which you want to paste the component, right-click the location in the flow chart where you want to paste it, and then select **Paste**.

4. Click  .

Fixed Loop Components

A fixed loop component is an object that cycles through a list of metadata members a fixed number of times.

Related Topics

- About Fixed Loop Components
- Creating a Fixed Loop Component
- Opening a Fixed Loop Component
- Editing a Fixed Loop Component
- Deleting a Fixed Loop Component
- Copying and Pasting a Fixed Loop Component

About Fixed Loop Components

A fixed loop component is an object that cycles through a list of metadata members a fixed number of times. For example, you can create a fixed loop that loops through a list of accounts ten times.

Creating a Fixed Loop Component

You create a fixed loop component in a business rule or template by dragging its icon and dropping it into the Rule Designer or Template Designer flow chart. A fixed loop component exists only within the business rule or template for which you create it. Fixed loop components cannot be shared across business rules or templates.

To create a fixed loop component:

1. In **System View**, right-click a rule or a template, and then select **Open**.
2. After you determine where in the business rule or template flow chart you want to create the fixed loop, from **New Objects**, drag the **Fixed Loop** object, and drop it into the flow chart.

   The fixed loop is displayed as two circles connected by a line.
3. **Custom template users only:** If you are creating a fixed loop component for a template, create design-time prompts for it on the **design-time prompt** tab. See Creating Design-Time Prompts for Custom Templates.
4. On the **Fixed Loop** tab, do these tasks:
   a. In **Value**, enter the number of times you want the loop to cycle through the metadata or data. Or click the **Actions** icon and select **Variable** (if you are working with a business rule) or **DTP** (if you are working with a custom template) or **Function** (if you are working with either a business rule or a custom template) to select variables, design-

time prompts, or functions to define the loop. See Working with Variables and Working with Functions.

b. In **Break Variable**, enter a variable, or click the **Variable** icon (if you are working with a business rule) or the **DTP** icon (if you are working with a custom template) to choose a numeric variable or a design-time prompt, respectively, to exit the fixed loop. The value of the break variable must be one.

c. On the **Errors and Warnings** tab, click the button to run script diagnostics.

5. **Optional:** On the **Properties** tab, do any of these tasks:
   - Select **Disabled** to exclude the fixed loop component from the business rule or template validation and calculation.
   - Enter a caption to identify the fixed loop component. The caption is displayed below the component in the flow chart of the rule or template to which it belongs.
   - Enter a description and comments for the fixed loop component.
6. Click  .

Opening a Fixed Loop Component

You open a fixed loop component from the flow chart of the business rule or template to which it belongs. Unlike formula and script components, you cannot open it from the System View.

To open a fixed loop component:

1. In **System View**, right-click a rule or a template, and then select **Open**.
2. When the rule or template opens, select the fixed loop component in the flow chart to see the fixed loop properties.

Editing a Fixed Loop Component

You can edit the value you assign to a fixed loop component and the variable you select for it. If you are creating a fixed loop for a business rule, you can also edit the caption and the break variable. A break variable specifies when to exit the fixed loop.

To edit a fixed loop component:

1. In **System View**, right-click the rule or template that contains the fixed loop component, and then select **Open**.
2. Click  .

Deleting a Fixed Loop Component

You delete a fixed loop component by removing it from the business rule or template to which it belongs. A fixed loop component can be used in only one business rule or template, so you delete it simply by removing it from the business rule or template.

To delete a fixed loop component:

1. In **System View**, right-click the rule or template that contains the fixed loop component to delete, and then select **Open**.
2. In the **Rule Designer** or **Template Designer**, select the fixed loop component you want to delete in the flow chart.

3. Right-click the fixed loop component, and then select **Remove**.
4. Confirm deletion of the component, and the click  .

Copying and Pasting a Fixed Loop Component

You can copy a fixed loop component from a rule or template and paste it into the same, or a different, rule or template. You cannot copy a fixed loop component and paste it into another fixed loop component or another component type.

To copy and paste a fixed loop component from the Rule Designer or Template Designer:

1. In **System View**, right-click the rule or template that contains the fixed loop component to copy, and then select **Open**.
2. In the **Rule Designer** or **Template Designer**, right-click the fixed loop component to copy in the flow chart, and then select **Copy**.
3. Do one of these tasks:
   - To paste the fixed loop component into the *same* business rule or template, right-click the location where you want to paste it in the flow chart, and then select **Paste**.
   - To paste the fixed loop component into a *different* business rule, open the business rule into which you want to paste the component, right-click the location in the flow chart where you want to paste the fixed loop component, and then select **Paste**.
4. Click  .

Metadata Loop Components

Metadata loop components enable you to assign a value to multiple members using either a function (metadata) or a start and end value (fixed).

Related Topics

- About Metadata Loop Components
- Creating Metadata Loop Components
- Opening Metadata Loop Components
- Deleting Metadata Loop Components
- Copying and Pasting Metadata Loop Components

About Metadata Loop Components

Metadata loop components enable you to assign a value to multiple members using either a function (Metadata) or a start and end value (Fixed).

For example, you may want to assign `#missing` to all the "level 0" accounts under the parent "Gross Profit". In this example, you would select the following in the metadata loop:

- Dimension—"Account"
- Function—`@RELATIVE("GP",0)`
- Variable—A "member" type design-time prompt

Creating Metadata Loop Components

You create a metadata loop component by dragging its icon and dropping it into the flow chart of a custom template. A metadata loop component exists only within the template for which you create it. You cannot share metadata loop components across templates.

To create a metadata loop component:

1. In **System View**, right-click a custom template, and then select **Open**.
2. In the **Template Designer**, under **New Objects**, select the **Metadata Loop** object, and then drop it into the flow chart.
3. In the **Metadata Loop** tab, select one of the following loop types types:
   - **Metadata**

     If you select **Metadata**, enter the following information:

     - **Index DTP**—Select an existing design-time prompt that functions as an index for the metadata loop. The value of the design-time prompt is reset to zero before the start of the metadata loop and increments by one for each loop.
     - **Dimension**—Select the dimension that contains the parent to use in the metatdata loop.
     - **Function**—Select a function to specify how to apply the variable to the members of the dimension.
     - **Variable**—Create a design-time prompt *member* to use for the variable. This design-time prompt is used in the formula, and Calculation Manager replaces the design-time prompt with each member created by the function specified.
   - **Fixed**

     If you select **Fixed**, enter the following information:

     - **Index DTP**—Select an existing step design-time prompt that functions as an index for the metadata loop. The value of the design-time prompt is reset to zero before the start of the metadata loop and increments by one for each loop.
     - **Start Index**—Value where the loop begins. Click  to select a design-time prompt to be used as the start index.
     - **End Index**—Value where the loop ends. Click  to select a design-time prompt to be used as the end index.
4. Enter properties for the metadata loop:
   - Select **Disabled** to exclude the metadata loop component from the template for validation and calculation purposes.

     You may want to disable a component when a business rule does not validate and you need to find the source of the validation error. When you disable a component, it is displayed in gray in the flow chart.
   - Enter a **Caption** for the metadata loop.

     The caption displays below the metadata loop component's icon in the flow chart.
   - Enter a **Description** and **Comments**.
5. Click  .

Opening Metadata Loop Components

You open a metadata loop component from the flow chart of the custom template to which it belongs. Unlike formula and script components, you cannot open a metadata loop component directly from **System View**.

To open a metadata loop component:

1. In **System View**, right-click a custom template, and then select **Open**.
2. In the **Template Designer,** select the metadata loop component in the flow chart to see its properties.

Deleting Metadata Loop Components

You can delete a metadata loop component from within the custom template to which it belongs.

To delete a metadata loop component:

1. In **System View**, right-click the custom template that contains the metadata loop, and then select **Open**.
2. In the **Template Designer**, in the flow chart, right-click the metadata loop component, and then select **Remove**.

> **Note**
>
> These steps only delete the metadata loop from the template. You should also delete the design-time prompts you created to use with the metadata loop. To delete a design-time prompt, from the **design-time prompt** tab, right-click the row containing the design-time prompt, and then select **Delete Row**, or select the row containing the design-time prompt, and then click  .

Copying and Pasting Metadata Loop Components

You can copy a metadata loop component from a custom template and paste it into either the same template or a different custom template.

To copy and paste a metadata loop component:

1. In **System View**, right-click the custom template that contains the metadata loop component, and then select **Open**.
2. In the **Template Designer**, in the flow chart, right-click the metadata loop component to copy, and then select **Copy**.

   If you dragged a formula or script component inside the metadata loop, select **Copy Group** to copy both the metadata loop and the component inside the metadata loop
3. Do one of these tasks:
   - To paste the metadata loop component into the *same* template, right-click the desired location in the flow chart, and then select **Paste**.
   - To paste the metadata loop component into a *different* template, open the desired template, then right-click the desired location in the flow chart, and then select **Paste**.

4. Click  .

> **Note**
>
> To copy design-time prompts if they are used in the metadata loop, in the **design-time prompt** tab, right-click the row containing the design-time prompts to copy, and then select **Copy**.
>
> To copy all design-time prompts, select **Copy Grid**, then open the template where you want to paste the design-time prompts, then in the **Design-Time Prompt** tab, click in the grid, and then right-click and select **Paste**.

DTP Assignment Components

Use a DTP Assignment Component to assign a DTP, member, function, or typed text to a design-time prompt in a custom template.

Related Topics

- About DTP Assignment Components
- Creating DTP Assignment Components
- Opening DTP Assignment Components
- Editing DTP Assignment Components
- Deleting DTP Assignment Components
- Copying and Pasting DTP Assignment Components
- Using Design-Time Prompt Functions in DTP Assignment Components

About DTP Assignment Components

You can assign a DTP, member, function, or typed in text to a design-time prompt in a custom template using a DTP Assignment component. DTP Assignment components reduce the complexity of the template's flow chart and makes the logic of the template easier to develop and maintain.

Creating DTP Assignment Components

You create a DTP Assignment component in a template by dragging its icon and dropping it into the Template Designer flow chart.

When creating DTP Assignment components, note the following:

- A DTP Assignment component exists only within the template for which you create it.
- You can place a DTP component anywhere in the flow chart except inside a member range, member block, or a condition component that does not use a DTP condition.

To create a DTP Assignment component:

1. In **System View**, open a custom template.
2. In the **Template Designer**, under **New Objects**, select the **DTP Assignment** object, and then drop it into the flow chart.

3. **Optional:** Enter design-time prompts for the component. See Creating Design-Time Prompts for Custom Templates.
4. In the **Formula** tab, define conditions for the template's design-time prompts.
   - From the dropdown to the left of the equal sign, select the design-time prompt for which you want to define a value.
     If there are no design-time prompts in the dropdown, none were created for the template with which you are working.
   - In the text box to the right of the equal sign, enter a value, or click  to select a variable, member, or function.
   - Click  to assign values for additional design-time prompts.
5. Enter properties for the DTP Assignment component:
   - Select **Disabled** to exclude the metadata loop component from the template for validation and calculation purposes.

     You may want to disable a component when a business rule does not validate and you need to find the source of the validation error. When you disable a component, it is displayed in gray in the flow chart.
   - Enter a **Caption** for the DTP Assignment component.

     The caption displays below the DTP Assignment component's icon in the flow chart
   - Enter a **Description** and **Comments**.
6. Click  .

Opening DTP Assignment Components

You open a DTP Assignment component from the flow chart of the template to which it belongs. Unlike formula and script components, you cannot open a DTP Assignment component directly from **System View**.

To open a DTP Assignment component:

1. In **System View**, right-click a custom template, and then select **Open**.
2. In the **Template Designer,** select the DTP Assignment component in the flow chart to see its properties.

Editing DTP Assignment Components

You can edit a DTP Assignment component's description, caption, and comments. You can also edit the values you assign to the design-time prompts in a template using the DTP Assignment component.

To edit a DTP Assignment component:

1. In **System View**, right-click the template that contains the DTP Assignment component to edit, and then select **Open**.
2. In the **Template Designer**, select the DTP Assignment component in the flow chart.
3. Make the desired changes, and then click  .

Deleting DTP Assignment Components

You delete a DTP Assignment component from within the template to which it belongs.

To delete a DTP Assignment component:

1. In **System View**, right-click the custom template that contains the DTP Assignment component, and then select **Open**.
2. In the **Template Designer**, in the flow chart, right-click the DTP Assignment component, and then select **Remove**.

Copying and Pasting DTP Assignment Components

You can copy a DTP assignment component from a template and paste it into either the same template or a different, template.

To copy and paste a DTP assignment component

1. In **System View**, right-click the custom template that contains the DTP assignment component, and then select **Open**.
2. In the **Template Designer**, in the flow chart, right-click the DTP Assignment component to copy, and then select **Copy**.
3. Do one of these tasks:
   - To paste the DTP Assignment component into the *same* template, right-click the desired location in the flow chart, and then select **Paste**.
   - To paste the DTP Assignment component into a *different* template, open the desired template, then right-click the desired location in the flow chart, and then select **Paste**.

     If you paste the DTP assignment into a new template, you must also create or copy the design-time prompts used by the DTP Assignment component into the new template.
4. Click  .

Using Design-Time Prompt Functions in DTP Assignment Components

Related Topics

- About Design-Time Prompt Functions
- @AvailDimCount
- @Compare
- @Compliment
- @Concat
- @DenseMember
- @Dependency
- @DimAttribute
- @DimMember
- @DimName

- @DimType
- @DimUDA
- @EndsWith
- @Evaluate
- @FindFirst
- @FindLast
- @GetData
- @Integer
- @Intersect
- @IsAncest
- @IsChild
- @IsDataMissing
- @IsSandBoxed
- @IsVariable
- @Length
- @Matches
- @Member
- @MemberGeneration
- @MemberLevel
- @MsgFormat
- @Notin
- @OpenDimCount
- @Plandim
- @PlanDimMember
- @Quote
- @RemoveQuote
- @ReplaceAll
- @ReplaceFirst
- @SmartListFromIndex
- @SmartListFromValue
- @SparseMember
- @StartsWith
- @SubString
- @ToLowerCase
- @ToMDX
- @ToUpperCase
- @Trim
- @Union

- @ValueDimCount

About Design-Time Prompt Functions

You can use design-time prompt functions in DTP Assignment components for custom templates. These functions enable you to perform tasks such as comparing a member to another member, getting a list of members that are common or not common between two design-time prompts, adding or removing quotes in a string, converting characters in a string to upper or lower case, and other tasks.

@AvailDimCount

`@AVAILDIMCOUNT` returns the number of available dimensions.

DTP Type:

`@AVAILDIMCOUNT` can be assigned to a DTP of type *numeric*.

Syntax:

`@AVAILDIMCOUNT(`*`DTP`*`,Dense/Sparse)`

Parameters:

Parameter	Description
DTP	Design-time prompt or function that returns a member range or cross dimension.
`Dense/Sparse`	**Optional.** If left empty, returns the count of all available dimensions. If "Dense" or "Sparse" is entered, only the specified type of available dimensions are counted.

Example 1:

```
[DTP1]=@AVAILDIMCOUNT([MR1])
```

Where:

- `[DTP1]` is a DTP of type *numeric* .
- `[MR1]` is a DTP of type *member range* with the following inputs:
  - Account:
  - Period: Jan, Feb, Mar
  - HSP_View:
  - Year:
  - Scenario:
  - Version: Working
  - Entity:
  - Product: P_000

In this example, `[DTP1]` returns "8", since there are a total of eight dimensions.

Example 2:

```
[DTP2]=@AVAILDIMCOUNT([MR2],"Dense")
```

Where:

- [DTP2] is a DTP of type *numeric*.
- [MR2] is a a DTP of type *member range* with the following inputs:
  - Account: Dense
  - Period: Dense
  - HSP_View: Sparse
  - Year: Sparse
  - Scenario: Sparse
  - Version: Sparse
  - Entity: Sparse
  - Product: Sparse

In this example, DTP2 returns "2", since only "Account" and "Period" are dense.

@Compare

@COMPARE returns "true" or "false" when comparing a member to a member, a dimension to a dimension, a password to password, or a string to a restricted list. (For a restricted list, @COMPARE compares using the rule builder value, not the substituted value.)

DTP Type:

@COMPARE can be assigned to a DTP of type of *boolean*.

Syntax:

```
@COMPARE(String,compareString,IgnoreCase)
```

Parameters:

Parameter	Description
String	Design-time prompt or text
compareString	Design-time prompt or text
IgnoreCase	**Optional**. If "true," the case of parameters 1 and 2 is ignored. If "false," the case must match for parameters 1 and 2. If *IgnoreCase* is left empty, it defaults to "true."

Example:

```
[DTP1]=@COMPARE([MBR1],[MBR2],true)
```

Where:

- [DPT1] is a non-promptable DTP of type *boolean*
- [MBR1]=Jan
- [MBR2]=Jan

In this example, [DTP1] returns "true."

@Compliment

`@COMPLIMENT` returns the members in DTP1 that are not in DTP2, and the members that are in DTP2 that are not in DTP1. In other words, `@COMPLIMENT` returns what is not common between DTP1 and DTP2.

DTP Type:

`@COMPLIMENT` can be assigned to a DTP of type *member range* or *cross dimension*.

Syntax:

`@COMPLIMENT(Argument1,Argument2)`

Parameters:

Parameter	Description
Argument1	Design-time prompt or function that returns a *member range* or *cross dimension*
Argument2	Design-time prompt or function that returns a *member range* or *cross dimension*

Example 1:

```
[DTP_MR]=@COMPLIMENT([MR1],[MR2])
```

Where:

- `[DTP_MR]` is a DTP of type *member range* that is not a checked prompt.
- `[MR1]`="Salaries,"FY14","Local","Tennessee","USA"
- `[MR2]`="Jan", "Feb", "Mar","FY15","Actual", "Budget","Working","Florida","California","000", "G100"

In this example, `DTP_MR` returns "Salaries,"Jan", "Feb", "Mar", "Actual", "Budget","Working","Local","000", "G100","USA"

Example 2:

```
[DTP_CD]=@COMPLIMENT([CD1],[CD2])
```

Where:

- `[DTP_CD]` is a DTP of type *cross dimension* that is not a checked prompt.
- `[CD1]`="Salaries"->"Jan"->"Actual"->"Local"
- `[CD2]`="50100"->"Feb"->"Budget"->"Working"->"California"->"000"

In this example, `[DTP_CD]` returns "Working"->"Local"->"California"->"000"

@Concat

`@CONCAT` concatenates the second parameter to the end of the first parameter.

DTP Type:

`@CONCAT` can be assigned to a DTP of type *member*, *dimension*, *string*, *restricted list*, *password*, or *attribute*.

Syntax:

`@CONCAT(Text, Concat String, Seperator)`

Parameters:

Parameter	Description
Text	Design-time prompt or text
Concat String	Design-time prompt or text **Note:** *concatString* will be added to the end of *String*
Seperator	**Optional:** If a separator is used, then the separator is added between the *Text* and *Concat String* only if *Text* has a value.

Example:

`[DTP_CTmbr]=@CONCAT([MBR1],[MBR2])`

Where:

- `[DTP_CTmbr]` is a DTP of type *member* that is not a checked prompt.
- `[MBR1]`="California"
- `[MBR2]`="Washington"

In this example, `[DTP_CTmbr]` returns "CaliforniaWashington".

@DenseMember

`@DENSEMEMBER` returns the first dense dimension member in a cross dimension.

DTP Type:

`@DENSEMEMBER` can be assigned to a DTP of type *member*, *members*, *dimension*, *dimensions*, *member range*, *cross dimension*, or *string*.

Syntax:

`@DENSEMEMBER(Members)`

Parameter:

Parameter	Description
Members	Design-time prompt, member, or function that returns a *cross dimension* or *member range*

Example:

`[DenseMbr]=@DENSEMEMBER([CD1])`

Where:

- `[DenseMbr]` is a DTP of type *cross dimension* that is not a checked prompt.
- `[CD1]` is a DTP of type *cross dimension* that is promptable.
- `[CD1]`="Salaries"->"Jan"->"Working"->"Tennessee"->"G401"

In this example, [Densembr] returns "Salaries", since "Salaries" is the first dense member in the cross dimension list.

@Dependency

"Inclusive" returns member(s) from Input 1 for which Input 2 has member(s) specified from the same dimensions. "Exclusive" returns members from Input 1 for which Input 2 has no specified members in the same dimensions.

Destination DTP types can be used with the following DTPs for @DEPENCENCY:

Destination DTP Types	Parameters to Use In @DEPENDENCY
Member	Member Range, Cross Dimension, Dimension, or Dimensions. If member is used in @DEPENDENCY, it must be used within the function @DIMMEMBER (member).
Members	Dimension or Dimensions
Dimension	Member Range, Cross Dimension, Dimension, or Dimensions
Dimensions	Member Range or Cross Dimension
Member Range	Member Range, Cross Dimension, Dimension, or Dimensions
Cross Dimension	Member Range, Cross Dimension, Dimension, or Dimensions

Syntax:

@DEPENENCY (*Input1, Dependence, Input2*)

Parameters:

Parameter	Description
Input1	design-time prompt or function
Dependence	"Inclusive" or "Exclusive"
Input2	design-time prompt or function

Example 1 and Example 2 below assume the following inputs:

- [cd_mr_i1] is a DTP of type *cross dimension* that is not a checked prompt.
- [CD1] is a promptable DTP of type *cross dimension* with the following inputs:
  - Account: "Units"
  - Period: "Jan"
  - HSP_View:
  - Year: "FY15"
  - Scenario: "Actual"
  - Version: "Working "
  - Entity: "111"
  - Product:
- [POV] is a promptable DTP *member range* with the following inputs:
  - Account:
  - Period:"Feb"

- HSP_View:
- Year: "FY16"
- Scenario: "Actual"
- Version: "Working "
- Entity:
- Product: "P_000"

Example 1:

```
[cd_mr_i1]=@DEPENDENCY([CD1],"Inclusive",[POV])
```

In this example, `[cd_mr_i1]` returns "Jan->FY15->Actual->Working" since "Period ", "Year", "Scenario", and "Version" have inputs for "CD1" and "POV".

Example 2:

```
[cd_mr_e1]=@DEPENDENCY([CD1],"Exclusive",[POV]
```

In this example, `[cd_mr_e1]` returns "Units->"111" " since "Account" and "Entity" are the only dimensions in "Input 1" that have members specified and that are not specified in "Input 2".

@DimAttribute

`@DIMATTRIBUTE` returns the attribute name if the specified attribute is associated with a dimension.

DTP Type:

`@DIMATTRIBUTE` can be assigned to a DTP of type *attribute*, *member*, *dimension*, or *string*.

Syntax:

`@DIMATTRIBUTE(`*`Dimension,Attribute`*`)`

Parameters:

Parameter	Description
`Dimension`	Design-time prompt or dimension drop-down that allows you to select dimensions from the application
`Attribute`	Design-time prompt, attribute (dimension or member), or function that returns an attribute dimension name or an attribute member name

Example:

```
[DIMA_ATTRB]=@DIMATTRIBUTE([DIM],[ATTRB])
```

Where:

- `[DIMA_ATTRB]` is a DTP of type *attribute* that is not a checked prompt.
- `[DIM]` is a promptable DTP of type *dimension*.
- `[ATTRB` is a promptable DTP of type *attribute*.

Assume the following values are input:

- `[DIM]`: "Entity"

- [ATTRB]: "Small"

In this example, since "Entity" has an attribute dimension named "Size", and "Small" is a member under "Size", [DIMA_ATTRB] returns "Small".

@DimMember

@DIMMEMBER returns a member if it is valid for the specified dimension.

DTP Type:

@DIMMEMBER can be assigned to a DTP of type *member*, *members*, *cross dimension*, *member range*, *dimension*, or *dimensions*.

Syntax:

@DIMMEMBER(*Dimension,Member*)

Parameters:

Parameter	Description
Dimension	Design-time prompt or dimension
Member	Design-time prompt, member, or function drop-down that shows the list of dimensions in the application from which to select

Example:

[DIM_mbr1]=@DIMMEMBER([DIM],[Member])

Where:

- [DIM_mbr1] is a DTP of type *member* that is not a checked prompt.
- [DIM] is a DTP of type *dimension* that is a checked prompt.
- [Member] is a DTP of type *member* that is a checked prompt.
- [Dim]="Entity"
- [Member]="Washington"

In this example, [DIM_mbr1] returns "Washington,"since Washington is a member of the Entity dimension.

@DimName

@DIMNAME returns the name of a dimension if it is valid for the database.

DTP Type:

@DIMNAME can be assigned to a DTP of type *dimension*.

Syntax:

@DIMNAME(*Dimension*)

Parameter:

Parameter	Description
Dimension	Design-time prompt, dimension, or typed in text. Click the dimension to display a list of available dimensions.

Example:

```
[DIMN_Product]=@DIMNAME(["Product"])
```

Where:

- `[DIMN_Product]` is a DTP of type *dimension* that is not checked prompt.
- `"Product"` is typed in the function.

If "Product" is a valid dimension name for this plan type, then `[DIMN_Product]` returns "Product".

If "Product" is not a valid dimension for this plan type, then `[DIMN_Product]` returns "empty".

@DimType

`@DIMTYPE` returns a dimension that matches the specified type.

DTP Type:

`@DIMTYPE` can be assigned to a DTP of type *dimension*, *dimensions*, or *string*.

Syntax:

`@DIMTYPE(`*`Dimension Type`*`)`

Parameter:

Parameter	Description
Dimension Type	Dimension types used in the application

Example:

```
[Dims]=@DIMTYPE(Account)
```

Where:

- `[Dims]` is a DTP of type *dimension* that is not a checked prompt.
- `[Account]` is selected from the drop down.

In this example, `[Dims]` returns "Account." If the Account dimension had been named Measures in the application, it would have returned "Measures."

@DimUDA

`@DIMUDA` returns the UDA name if the specified UDA is valid for the dimension.

DTP Type:

`@DIMUDA` can be assigned to a DTP of type *UDA*, *Member*, *Dimension*, or *String*.

Syntax:

`@UDA(Dimension,UDA)`

Parameters:

Parameter	Description
Dimension	Design-time prompt or dimension Click the dimension to display a list of available dimensions.
UDA	If you select a DTP or UDA in the *Dimension* parameter, then select a UDA from the drop-down list.

Example:

```
[dimu_mbr2]=@DIMUDA([DIM],[UDA2] [dimu_mbr]=@DIMUDA([DIM],[UDA]
```

Where:

- `[dimu_mbr2]` is a DTP of type *member* that is not a checked prompt.
- `[dimu_mbr]` is a DTP of type *member* that is not a checked prompt.
- `[dim]` is a DTP of type *dimension* that is a checked prompt
- `[UDA2]` is a DTP of type *UDA* that is a checked prompt.
- `[UDA]` is a DTP of type *UDA* that is a checked prompt.

Assume that the following values are input:

- `[dim]`: "Account"
- `[UDA2]`: "HSP_LEAPYEAR"
- `[UDA]`: "Revenue"

In this example, `[dimu_mbr2]` returns nothing, since "HSP_LEAPYEAR" is not a UDA on "Account," and `[dimu_mbr]` returns "Revenue."

@EndsWith

`@ENDSWITH` tests to see if the string ends with the specified suffix.

DTP Type:

`@ENDSWITH` can be assigned to a DTP of type *boolean*.

Syntax:

`@ENDSWITH(String,Suffix)`

Parameters:

Parameter	Description
String	Design-time prompt or text

Parameter	Description
Suffix	Design-time prompt or text

Example:

```
[DTP_end]=@ENDSWITH([Mbr1],[String1])
```

Where:

- `[Mbr1]` is a DTP of type *member* that is a checked prompt.
- `String1` is a DTP of type *string* that is a checked prompt.

Assume that the following values are input:

- `[Mbr1]`: "Joe Smith"
- `[String1]`: "Smith"

In this example, `[DTP_end]` returns "true."

@Evaluate

`@EVALUATE` returns the result of an expression.

DTP Type:

`@Evaluate` can be assigned to a DTP of type *numeric* or *integer*.

Syntax:

`@EVALUATE(Value1,Operator,Value2)`

Parameters:

Parameter	Description
Value1	Design-time prompt or function that returns a numeric value
Operator	+, -, /, or *
Value2	Design-time prompt or function that returns a numeric value

Example:

```
[Eval_num_add]=@EVALUATE([num1], "+",[num2])
```

Where:

- `[Eval_numadd]` is a DTP of type *numeric* that is not checked prompt.
- `[num1]` is a promptable DTP of type *numeric*.
- The operator is +.
- `[num2]` is a promptable DTP of type *numeric*.

Assume that the following values are input.

- `[num1]`: "100"
- `[num2]`: "10"

In this example, [Eval_numadd] returns "110.0".

@FindFirst

@FINDFIRST finds the first substring of a string that matches the given regular expression.

DTP Type:

@FINDFIRST can be assigned to a DTP of type *string*, *password*, *member*, *members*, *dimension*, or *dimensions*.

Syntax:

@FINDFIRST(*text*,*regExpr*,*IgnoreCase*)

Parameters:

Parameter	Description
text	Design-time prompt, member, function, or typed in text
regExpr	See Java docs for "java.util.regex.Pattern."
IgnoreCase	**Optional.** True or False. If left empty, defaults to True.

Example:

[FINDF_PW_T]=@FINDFIRST([PW],[FindF_String_PW],"true")

Where:

- [FINDF_PW_T] is a DTP of type *password* that is not a checked prompt.
- [PW] is a promptable DTP of type *password*.
- [FindF_String_PW] is a promptable DTP *string*.

Assume that the following values are input:

- [PW]: " test20pw15test"
- [FindF_String_PW]: "\d\d " (which is the JAVA expression to return any digit followed by a digit)
- Ignorecase: "true"

In this example, [FINDF_PW_T] returns "20".

@FindLast

@FINDLAST finds the last substring of a string that matches the given regular expression.

DTP Type:

@FINDLAST can be assigned to a DTP of type *string*, *password*, *member*, *members*, *dimension*, or *dimensions*.

Syntax:

@FINDLAST(*text*,*regExpr*,*IgnoreCase*)

Parameters:

Parameter	Description
text	Design-time prompt, member, function, or typed in text.
regExpr	See Java docs for "java.util.regex.Pattern."
IgnoreCase	**Optional.** True or False. If left empty, defaults to True.

Example:

```
[FINDL_PW_T]= @FINDLAST([PW],[FindL_String_PW],"true")
```

Where:

- `[FINDL_PW_T]` is a DTP of type *password* that is not a checked prompt.
- `[PW]` is a promptable DTP of type *password*.
- `[FindL_String_PW]` is a promptable DTP of type *password*.

Assume that the following values are input:

- `[PW]`: " test20pw15test"
- `[FindL_String_PW]`: "\d\d " (which is the JAVA expression to return any digit followed by a digit)
- `Ignorecase`: "true"

In this example, `[FINDL_PW_T]` returns "15".

@GetData

`@GETDATA` returns the value of the slice.

DTP Type:

`@GETDATA` can be assigned to a DTP of type *numeric*.

Syntax:

`@GETDATA(`*Members*`)`

Parameter:

Parameter	Description
Members	Design-time prompt, member, or function that returns a cross dimension

Example :

```
@GETDATA([CD]) < 10
```

Where the function is used on a step in a template for the enabling condition.

If the members entered for the promptable design-time prompt CD have a value of less than 10, then the step will display. If the member's value for the design-time prompt CD is greater than 10, then the step will not display

@Integer

`@INTEGER` returns an integer.

DTP Type:

`@INTEGER` can be assigned to a DTP of type *integer*.

Syntax:

`@INTEGER(value)`

Parameter:

Parameter	Description
value	Design-time prompt or function that returns a value

Example:

`[INT_NUM]=@INTEGER([num1])`

Where:

- `[INT_NUM]` is a DTP of type *integer* that is not a checked prompt.
- `[num1]` is a DTP of type *numeric* that is a checked prompt.

Assume that the input for `[num1]` is "15.7"

In this example, `[INT_NUM]` returns 15.

@Intersect

`@INTERSECT` returns members that are from common dimensions.

DTP Type:

`@INTERSECT` can be assigned to a DTP of type *member range* or *cross dimension*.

Syntax:

`@INTERSECT(Argument1,Argument2)`

Parameters:

Parameter	Description
Argument1	Design-time prompt, function, or text
Argument2	Design-time prompt, function, or text

Example:

`[IntersetMR]=@INTERSECT([MR1],[MR2])`

Where:

- `[MR1]` is a DTP of type *member range* that is a checked prompt.

- [MR2] is a DTP of type *member range* that is a checked prompt.

Assume that the input for MR1 is:

- Account: "Salaries"
- Version: "Target"
- Entity: "Tennessee", "Florida"

And the input for MR2 is:

- Account: "50590", "50350"
- Years: "FY10"
- Scenario: "Budget"
- Version: "First Pass"

In this example, [IntersectMR] returns "50590","50350","Salaries","First Pass","Target". Since members for "Entity" are only entered in MR1, and members for "Scenario" are only entered in MR2, these members are not included in [IntersectMR].

@IsAncest

@ISANCEST returns *true* if the specified ancestor member is an ancestor of the child member.

DTP Type:

@ISANCEST can be assigned to a DTP of type *boolean*.

Syntax:

@ISANCEST(*Ancestor Member/Child Member*)

Parameters:

Parameter	Description
Ancestor Member	Design-time prompt of type *member*
Child Member	Design-time prompt of type *member*

Example 1:

```
[True_False]=@ISANCEST([Anc_Mbr],[mbr])
```

Where:

- [True_False] is a DTP of type *boolean* that is not a checked prompt.
- [Prt_Mbr] is a promptable DTP of type *member type*.
- [Mbr] is a promptable DTP of type *member type*.

Assume the following:

- The input for [Anc_Mbr] is "Q1".
- The input for [Mbr1] is "Apr".

In this example, [True_False] returns *false*.

@IsChild

`@ISCHILD` returns *true* if the specified child member is child of the specified parent member.

DTP Type:

`@ISCHILD` can be assigned to a DTP of type *boolean*.

Syntax:

`@ISCHILD(Parent Member/Child Member)`

Parameters:

Parameter	Description
Parent Member	Design-time prompt of type *member*
Child Member	Design-time prompt of type *member*

Example 1:

```
[True_False]=@ISCHILD([Prt_Mbr],[mbr])
```

Where:

- `[True_False]` is a DTP of type *boolean* that is not a checked prompt.
- `[Prt_Mbr]` is a promptable DTP of type *member type*.
- `[Mbr]` is a promptable DTP of type *member type*.

Assume the following:

- The input for `[Prt_Mbr]` is "Q1".
- The input for `[Mbr]` is "Jan".

In this example, `[True_False]` returns *true*.

@IsDataMissing

`@ISDATAMISSING` returns true if the value of the slice is missing.

DTP Type:

`@ISDATAMISSING` can be assigned to a DTP of type *boolean*.

Syntax:

`@ISDATAMISSING(Members)`

Parameters:

Parameter	Description
members	Design-time prompt, member, or function that returns a cross dimension

Example:

```
@ISDATAMISSING([CD])=false
```

Where the function is used on a step in a tempate for the enabling condition.

If the members entered for the promptable design-time prompt CD have a value, then the step will display. if the member's value is `#Missing`, then the step will not display

@IsSandBoxed

`@ISSANDBOXED` determines if the current application is sandboxed.

DTP Type:

`@ISSANDBOXED` can be assigned to a DTP of type *boolean*.

Syntax:

`@ISSANDBOXED(`*`CubeName`*`)`

Parameter:

Parameter	Description
`CubeName`	Design-time prompt or text

Example:

```
[Sand1]=@ISSANDBOXED([cube1])
```

Where:

- `[Sand1]` is a DTP of type *boolean* that is not a checked prompt.
- `[cube1]` is a promtable DTP of type *string*.

Assume `[cube1]:Plan1`. In this example, `[Sand1]` returns *true* if the cube "Plan1 " is enabled for sandbox, and returns *false* if the cube "Plan1 " is not enabled for sandbox.

@IsVariable

`@ISVARIABLE` determines if the argument is a variable.

DTP Type:

`@ISVARIABLE` can be assigned to a DTP of type *boolean*.

Syntax:

`@ISVARIABLE(`*`Argument`*`)`

Parameters:

Parameter	Description
`Argument`	DTP types: Member, Members, String, Numeric, Percent, Integer, StringAsNumber, DateAsNumber

Example:

`[isVar_mbr]=@ISVARIABLE([Mbr1])`

Where:

- `[isVar_mbr]` is a DTP of type *boolean* that is not a checked prompt.
- `[Mbr1]` is a promptable DTP of type *member type*.

Assume that the input for `[Mbr1]` is "{Version}".

In this example, `[isVar_mbr]` returns "true".

@Length

`@LENGTH` returns the length of a string of text.

DTP Type:

`@LENGTH` can be assigned to a DTP of type *numeric* or *integer*.

Syntax:

`@LENGTH(Text)`

Parameters:

Parameter	Description
Text	Design-time prompt of type *string*

Example:

`[Len1]=@LENGTH([String1])`

Where:

- `[Len1]` is a numeric design-time prompt that is not checked prompt
- `[String1]` is a promptable design-time prompt type string

Assume that the following value is input:

`[String1]:` Today is the first day of fall

In this example, `Len1=30`

@Matches

`@MATCHES` returns "true" if the first substring of a string matches the given regular expression.

DTP Type:

`@MATCHES` can be assigned to a DTP of type *string*, *password*, *member*, *members*, *dimension*, or *dimensions*.

Syntax:

`@MATCHES(text,regExpr,IgnoreCase)`

Parameters:

Parameter	Description
text	Design-time prompt
regExpr	See Java docs for "java.util.regex.Pattern"
IgnoreCase	**Optional.** True or False. If left empty, defaults to True.

Example:

```
[Matches_Mbr]=@MATCHES([Mbr],[Match_String_Mbr],"true")
```

Where:

- `[Mbr]` is a promptable DTP *member type*.
- `[Match_String_Mbr]` is a promptable DTP *string.*
- `IgnoreCase` is "true".

Assume that the following values are input:

- `[Mbr]`: "P_100"
- `[Match_String_Mbr]` "\p{Alnum}", which is the Java regular expression to return an alphanumeric character.

In this example, `[Matches_Mbr]` returns "true".

@Member

`@MEMBER` returns a member, as long as it is a valid member.

DTP Type:

`@MEMBER` can be assigned to a DTP of type *member*.

Syntax:

`@MEMBER(`*Member*`)`

Parameter:

Parameter	Description
Member	Design-time prompt, member, or another design-time prompt function

Example:

```
[Mbr_mr1]=@UNION([MR1],@MEMBER([Member]))
```

Where:

- `[Mbr_mr1]` is a DTP of type *member range* that is not a checked prompt.
- `[@UNION]` creates the range of the DTP, MR1, and DTP Member as long as the member entered is a valid member.
- `[MR1]` is a DTP of type *member range* that is a checked prompt.
- `[Member]` is a DTP of type *member* that is a checked prompt.

Assume that the following values are input:

- `[MR1]`: Account "Salaries", Scenario "Actual", Entity "Tennessee"
- `[Member]`: "000", which is a member from the product dimension

In this example, `[Mbr_mr1]` returns "Salaries","Actual","Tennessee","000"

@MemberGeneration

`@MEMBERGENERATION` returns the generation number of a member.

DTP Type:

`@MEMBERGENERATION` can be assigned to a DTP of type *numeric* or *integer*.

Syntax:

`@MEMBERGENERATON(`*`Member`*`)`

Parameters:

Parameter	Description
`Member`	Design-time prompt, member, or function (design-time prompt or function should return a single member)

Example:

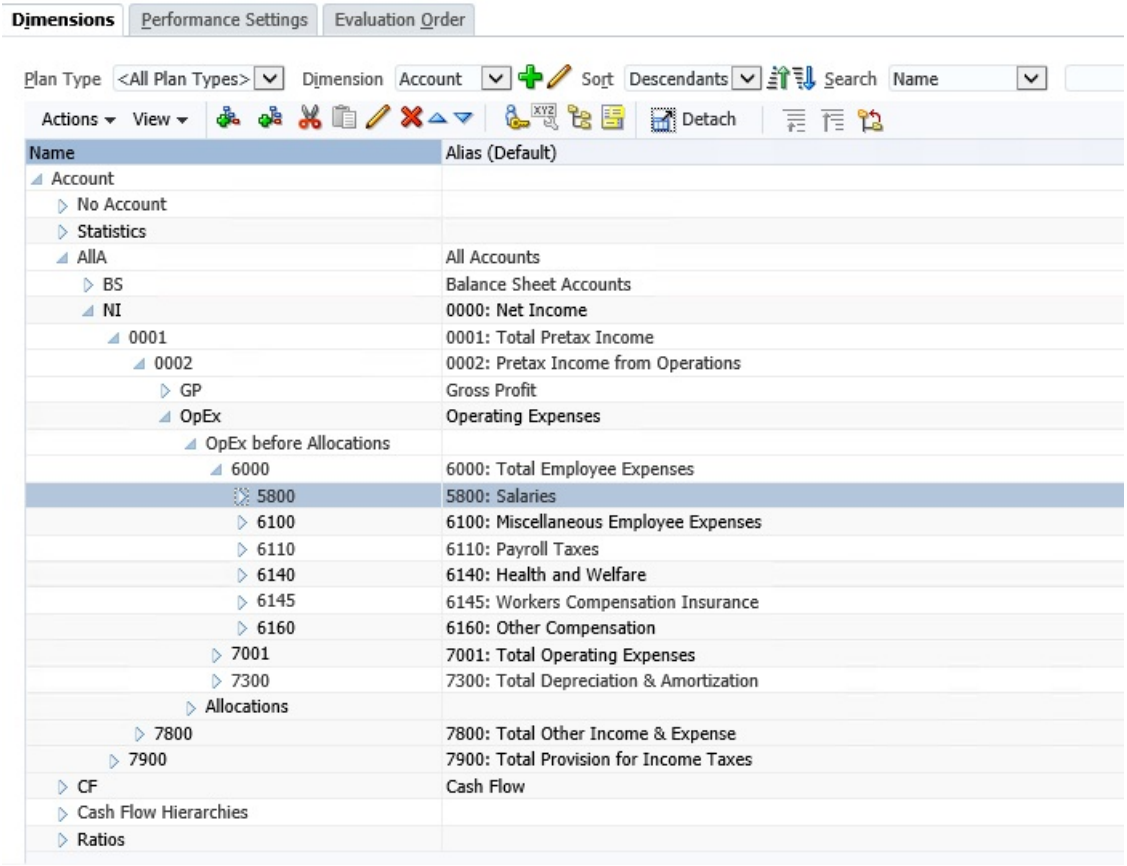
`[MBRGEN1]=@MEMBERGENERATION([member])`

Where:

- `[MBRGEN1]` is a DTP of type *numeric* that is not a checked prompt.
- `[member]` is a DTP of type *member* that is a checked prompt.

Assume that the input for `[member]` is "5800".

Using the following planning outline, `[MBRGEN1]` returns "9".



@MemberLevel

`@MEMBERLEVEL` returns the level number of a member.

DTP Type:

`@MEMBERLEVEL` can be assigned to a DTP of type *numeric* or *integer*.

Syntax:

`@MEMBERLEVEL(`*`Member`*`)`

Parameters:

Parameter	Description
`Member`	Design-time prompt, member, or function (design-time prompt or function should return a single member)

Example:

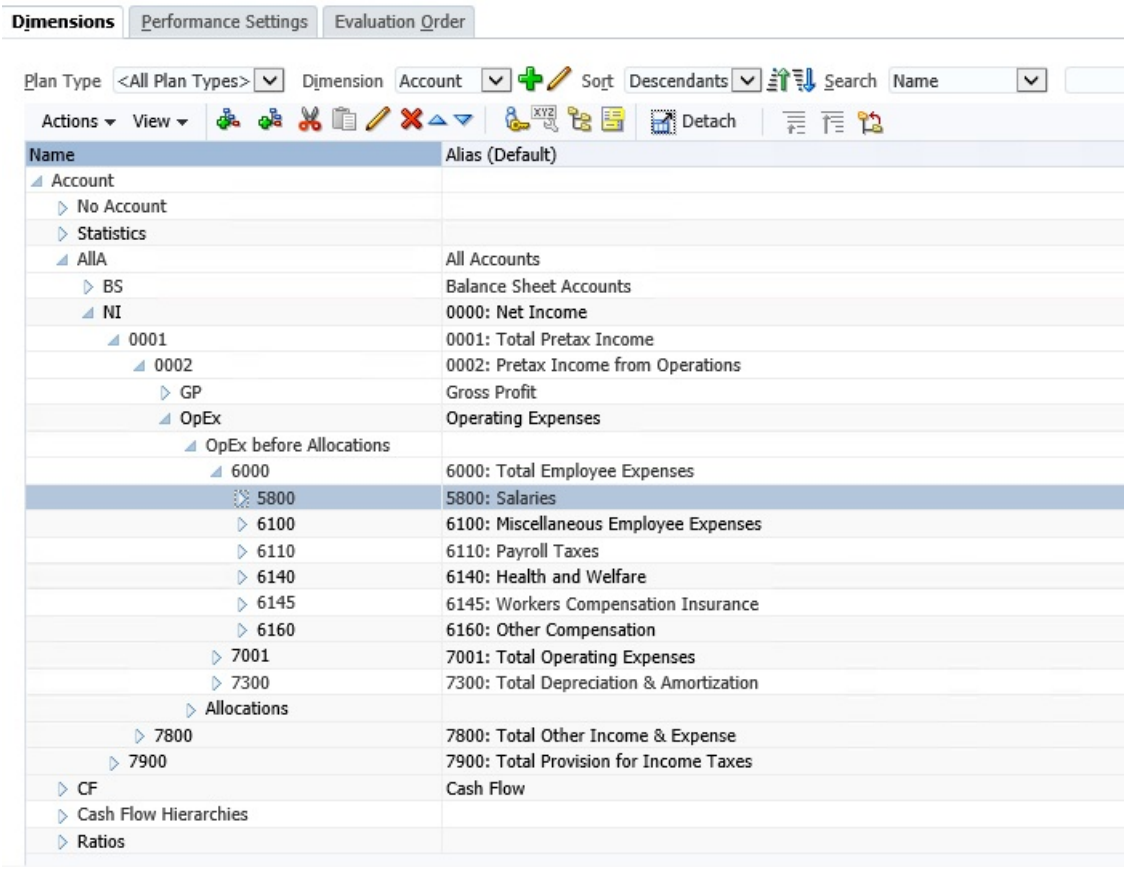
```
[MBRLEV_MBR]=@MEMBERLEVEL([member])
```

Where:

- `[MBRLEV_MBR]` is a DTP of type *numeric* that is not a checked prompt.
- `[member]` is a DTP of type *member* that is a checked prompt.

Assume that the input for `[member]` is "5800".

Using the following planning outline, `[MBRLEV_MBR]` returns 0.



@MsgFormat

`@MSGFORMAT` takes a set of objects, formats them, and then inserts the formatted strings into the pattern at the appropriate places. (See the JAVA docs for "java.text.MessageFormat.)"

DTP Type:

`@MSGFORMAT` must be *string*.

Syntax:

`@MSGFORMAT(`*`text,param1,param2,param3,param4`*`)`

Parameters:

Parameter	Description
`text`	Design-time prompt *string*
`param1`	Design-time prompt
`param2`	Design-time prompt
`param3`	Design-time prompt
`param4`	Design-time prompt

Example:

```
[MSFG4]=@MSGFORMAT([String4],[FirsName],[LastName],[Month],[Year])
```

Where:

- `[MSFG4]` is DTP *string* that is not a checked prompt.
- `[String4]` is a promptable DTP *string*.
- `[FirstName]` is a promptable DTP *string*.
- `[LastName]` is a promptable DTP *string*.
- `[Month]` is a promptable DTP *member* with "Period" selected for the dimension.
- `[Year]` is a promptable DTP *member* with "Year" selected for the dimension

Assume that the following values are input:

- `[String4]`: "{0} {1} completed this task in {2} {3}"
- `[FirstName]`: "Joe"
- `[LastName]`: "Smith "
- `[Month]`: "Feb"
- `[Year]`: "FY15"

In this example, `[MSFG4]` returns "Joe Smith completed this task in Feb "FY15."

@Notin

`@NOTIN` returns members that are in one expression, but not in another other expression. In other words, `@NOTIN` returns what is not common between the expressions.

DTP Type:

`@NOTIN` can be assigned to a DTP of type *member range* or *cross dimension*.

Syntax:

`@NOTIN(`*`Argument1,Argument2`*`)`

Parameters:

Parameter	Description
`Argument1`	Design-time prompt, function, or members
`Argument2`	Design-time prompt, function, or members

Example:

```
[NOTIN1]=@NOTIN([MR1],[MR2])
```

Where:

- `[NOTIN1]` is a DTP of type *member range* that is not a checked prompt.
- `[MR1]` is a DTP of type *member range* that is a checked prompt.
- `[MR2]` is a DTP of type *member range* that is a checked prompt.

Assume that the input for MR1 is:

- Period: "Jan", "Feb", "Mar"
- Scenario: "Actual"
- Currency: "Local"
- Product: "000"

Assume that the input for MR2 is:

- Period: "Feb"
- Version: "Working"
- Currency: "Local"
- Product: "000"

In this example, [@NOTIN1] returns "Actual" because this is the only member that is not in MR2, but is in MR1.

@OpenDimCount

@OPENDIMCOUNT returns the number of dimensions for which a member was not specified.

DTP Type:

@OPENDIMCOUNT can be assigned to a DTP of type *numeric*.

Syntax:

@OPENDIMCOUNT(*DTP*,Dense/Sparse)

Parameters:

Parameter	Description
DTP	Design-time prompt or function that returns a member range or cross dimension.
Dense/Sparse	**Optional.** If left empty, returns the count of all available dimensions that do not have a member specified. If "Dense" or "Sparse" is entered, only the specified type of dimensions that do not have members are counted.

Example 1:

```
[DTP1]=@OPENDIMCOUNT([MR1])
```

Where:

- [DTP1] is a DTP of type *numeric*.
- [MR1] is a DTP of type *member range* with the following inputs:
  - Account:
  - Period: Jan, Feb, Mar
  - HSP_View:
  - Year:
  - Scenario:
  - Version: Working

  - Entity: Sparse
  - Product: P-100

In this example, `[DTP1]` returns "5", since there are a total of eight dimensions and only three have member input.

Example 2:

```
[DTP2]=@OPENDIMCOUNT([MR1],"Dense")
```

Where:

- `[DTP2]` is a DTP of type *numeric*.
- `[MR1]` is a DTP of type *member range* with the following inputs:
  - Account: Dense
  - Period: Dense
  - HSP_View: Sparse
  - Year: Sparse
  - Scenario: Sparse
  - Version: Sparse
  - Entity: Sparse
  - Product: Sparse

In this example, `[DTP2]` returns "1", since "Account" and "Period" are dense, and "Period" has member input.

@Plandim

`@PLANDIM` returns the dimension name if it exists in the database of an application.

DTP Type:

`@PLANDIM` can be assigned to a DTP of type *member* or *dimension*. `@PLANDIM` works if only one value is used for the second parameter

Syntax:

`@PLANDMIN(`*`Database Name,Dimension`*`)`

Parameters:

Parameter	Description
`Database Name`	Design-time prompt or text
`Dimension`	Design-time prompt, dimension, or text

Example:

```
[PDIM_mbr]=@PLANDIM([PlTy],[Dim])
```

Where:

- `[PDIM_mbr]` is a DTP of type *member* that is not a checked prompt.

- `[PlTy]` is a DTP of type *string* that is a checked prompt.
- `[Dim]` is a DTP of type *dimension* that is a checked prompt.

Assume that the following values are input:

- `[PlTy]`: "Plan1"
- `Dim`: "Product"

In this example, `[PDIM_mbr]` returns "Product," because Product is a valid dimension in "Plan1".

@PlanDimMember

`@PLANDIMMEMBER` returns a member if the member is valid for the specified dimension in the specified plan type.

DTP Type:

`@PLANDIMMEMBER` can be assigned to a DTP of type *member*.

Syntax:

`@PLANDIMMEMBER(`*`plan type`*`,`*`dimension`*`,`*`member`*`)`

Parameters:

Parameter	Description
`plan type`	Design-time prompt that returns the type of plan
`dimension`	Design-time prompt that returns the dimension name
`member`	Design-time prompt that returns the member name

Example 1:

```
[DIM_mbr1]=@ PLANDIMMEMBER("Plan1",[DIM],[Member])
```

Where:

- `[DIM_mbr1]` is a DTP of type *member* that is not a checked prompt.
- `[DIM]` is a DTP of type *dimension* that is a checked prompt.
- `[Member]` is a DTP of type *member* that is a checked prompt.
- `[Dim]`="Entity"
- `[Member]`="Washington"

@Quote

`@QUOTE` adds double quotes around a string.

DTP Type:

`@QUOTE` can be assigned to a DTP of type *member*, *string*, or *password*.

Syntax:

`@QUOTE(`*`String`*`)`

Parameter:

Parameter	Description
String	Design-time prompt or text

Example:

```
[Quote_St]=@QUOTE([String1])
```

Where:

- `[Quote_St]` is a DTP of type *string* that is not a checked prompt.
- `[String1]` is a DTP of type *string* that is a checked prompt.

Assume that the input for `[String1]` is "Yellow".

In this example, `[Quote_St]` returns "Yellow".

@RemoveQuote

`@REMOVEQUOTE` removes a double quote from a string.

DTP Type:

`@REMOVEQUOTE` can be assigned to a DTP of type *member*, *string*, or *password*.

Syntax:

```
@REMOVEQUOTE(String)
```

Parameter:

Parameter	Description
String	Design-time prompt or text

Example:

```
[RemoveQuote_St]=@REMOVEQUOTE([String1])
```

Where:

- `[RemoveQuote_St]` is a DTP of type *string* and is not a checked prompt.
- `[String1]` is a DTP of type *string* that is a checked prompt.

Assume that the input for `[String1]` is "Yellow".

In this example, `[RemoveQuote_St]` returns "Yellow".

@ReplaceAll

`@REPLACEALL` replaces the part of a string that contains the expression with a replacement string.

DTP Type:

`@REPLACEALL` cab be assigned to a DTP of type *string*, *password*, *member*, *members*, *dimension*, or *dimensions*.

Syntax:

`@REPLACEALL(String,Regular Expression,replaceString)`

Parameters:

Parameter	Description
`String`	Design-time prompt or text
`Regular Expression`	Design-time prompt or text
`replaceString`	Design-time prompt or text

Example:

```
[ReplaceAll_Str]=@REPLACEALL([String1],[Rep_String],[new_String])
```

Where:

- `[ReplaceAll_Str]` is a a DPT of type *string* that is not a checked prompt.
- `[String1]` is a DTP of type *string* that is a checked prompt.
- `[Rep_String]` is a DTP of type *string* that is a checked prompt.
- `[new_String]` is a DTP of type *string* that is a checked prompt.

Assume that the parameters are as follows:

- String: "This is a test of a testing string"
- Regular Expression: "Test"
- Replace String "XYZ"

In this example, `[ReplaceAll_Str]` returns: "This is a XYZ of a XYZing string for one XYZ and two XYZ". It replaced with "test with XYZ. "

@ReplaceFirst

`@REPLACEFIRST` replaces the first occurrence of the regular expression with the replacement string.

DTP Type:

`@REPLACEFIRST` can be assigned to a DTP of type *string*, *password*, *member*, *members*, *dimension*, or *dimensions*.

Snytax:

`@REPLACEFIRST(String,Regular Expression,replaceString)`

Parameters:

Parameter	Description
String	Design-time prompt or text
Regular Expression	Design-time prompt or text
replaceString	Design-time prompt or text

Example:

```
[ReplaceFirst_Str]=@REPLACEFIRST([String1],[Rep_String],[new_String])
```

Where:

- `[ReplaceFirst_Str]` is a DPT of type *string* that is not a checked prompt.
- `[String1]` is a DTP of type *string* that is a checked prompt.
- `[Rep_String]` is a DTP of type *string* that is a checked prompt.
- `[new_String]` is a DTP of type *string* that is a checked prompt.

Assume that the parameters are as follows:

- String: "This is a testing of the new test of a test today"
- Regular Expression: "Test"
- Replace String: "XYZ"

In this example, `[ReplaceFirst_Str]` returns: "This is a XYZing of the new test of a test today". It will only replace the *first* occurrence of test, not all of the occurrences.

@SmartListFromIndex

`@SMARTLISTFROMINDEX` returns a Smart List member based on the order of the members in the SmartList.

DTP Type:

`@SMARTLISTFROMINDEX` can be assigned to a DTP of type *string*, *password*, *member*, or *members* .

Syntax:

`@SMARTLISTFROMINDEX(`*`SmartList,Index`*`)`

Parameters:

Parameter	Description
SmartList	Design-time prompt or dimension. The dimension shows the Smart List dimensions in the application.
Index	Design-time prompt or text

Example:

```
[sl_ind_Str]=@SmartListFromIndex([String1],[index_ST])
```

Where:

- `[sl_ind_Str]` is a DTP of type *string* that is not a checked prompt.
- `[String1]` is a DTP of type *string* that is a checked prompt.
- `[index_ST]` is a DTP of type *numeric* that is a checked prompt.

Assume that the parameters are as follows:

- SmartList: "CreditRatings"
- Index: "2"

In this example, the Smart List "CreditRatings" has the following data:

Name	Label
AA	ID_ENUMNETLABEL_AA
AAA	ID_ENUMNETLABEL_AAA
AAMinus	ID_ENUMNETLABEL_AA22
A	ID_ENUMNETLABEL_A
AMinus	ID_ENUMNETLABEL_AA22
AAPlus	ID_ENUMNETLABEL_AA11
APlus	ID_ENUMNETLABEL_A11
B	ID_ENUMNETLABEL_B
BPlus	ID_ENUMNETLABEL_B11
BMinus	ID_ENUMNETLABEL_B22
BB	ID_ENUMNETLABEL_BB
BBPlus	ID_ENUMNETLABEL_BB11
BBMinus	ID_ENUMNETLABEL_BB22
BBB	ID_ENUMNETLABEL_BBB
AAAMinus	ID_ENUMNETLABEL_AAA-
BBBPlus	ID_ENUMNETLABEL_BBB+
BBBMinus	ID_ENUMNETLABEL_BBB-
CCCPlus	ID_ENUMNETLABEL_CCC+
CCC	ID_ENUMNETLABEL_CCC

`[sl_ind_Str]` returns "AAMinus", because that is the name of the Smart List member at the second index position.

@SmartListFromValue

`@SMARTLISTFROMVALUE` returns a Smart List member based on the ID of the member in the SmartList.

DTP Type:

`@SMARTLISTFROMVALUE` can be assigned to a DTP of type *string,*, *password*, *member*, or *members* if only one member is entered.

Syntax:

`@SMARTLISTFROMVALUE(SmartList,Value)`

Parameters:

Parameter	Description
Smartlist	Design-time prompt or dimension. The dimension shows the Smart List dimensions in the application.
Value	Design-time prompt or number

Example:

```
[sl_val_Str]=@SmartListFromValue([String1],[value_ST])
```

Where:

- `[sl_val_Str]` is a DTP of type *string* that is not a checked prompt.
- `[String1]` is a DTP of type *string* that is a checked prompt.
- `[value_ST]` is a DTP of type *numeric* that is a checked prompt.

Assume that the parameters are as follows:

- Smart List: "CreditRatings"
- Value: "5"

In this example, the SmartList "CreditRatings" has the following data:

Name	Label
AA	ID_ENUMNETLABEL_AA
AAA	ID_ENUMNETLABEL_AAA
AAMinus	ID_ENUMNETLABEL_AA22
A	ID_ENUMNETLABEL_A
AMinus	ID_ENUMNETLABEL_A22
AAPlus	ID_ENUMNETLABEL_AA11
APlus	ID_ENUMNETLABEL_A11
B	ID_ENUMNETLABEL_B
BPlus	ID_ENUMNETLABEL_B11
BMinus	ID_ENUMNETLABEL_B22
BB	ID_ENUMNETLABEL_BB
BBPlus	ID_ENUMNETLABEL_BB11
BBMinus	ID_ENUMNETLABEL_BB22
BBB	ID_ENUMNETLABEL_BBB
AAAMinus	ID_ENUMNETLABEL_AAA-
BBBPlus	ID_ENUMNETLABEL_BBB+
BBBMinus	ID_ENUMNETLABEL_BBB-
CCCPlus	ID_ENUMNETLABEL_CCC+
CCC	ID_ENUMNETLABEL_CCC

`[sl_val_Str]` returns "AMinus" because that is the name of the SmartList member with the ID of 5.

@SparseMember

`@SPARSEMEMBER` returns the first sparse dimension member in a cross dimension.

DTP Type:

`@SPARSEMEMBER` can be assigned to a DTP of type *member*, *members*, *dimension*, *dimensions*, *member range*, *cross dimension*, or *string*.

Syntax:

`@SPARSEMEMBER(Members)`

Parameter:

Parameter	Description
Members	Design-time prompt, member, or function that returns a list of members

Example:

`[SparseMbr]=@SPARSEMEMBER([CD1])`

Where:

- `[SparseMbr]` is a DTP of type *cross dimension* that is not a checked prompt.
- `[CD1]` is a DTP of type *cross dimension* that is promptable.
- `[CD1]` is "Gross Sales"->"Jan"->"FY14"->"Working"->"California"

In this example, `[Sparsembr]` returns "FY14", since FY14 is the first sparse member in the cross dimension list.

@StartsWith

`@STARTSWITH` tests to see if the string starts with the specified prefix.

DTP Type:

`@STARTSWITH` can be assigned to a DTP of type *boolean*.

Syntax:

`@STARTSWITH(String,Prefix)`

Parameters:

Parameter	Description
String	Design-time prompt or text
Prefix	Design-time prompt or text

Example:

`[DTP_Start]=@STARTSWITH([Mbr1],[String1])`

Where:

- `[Mbr1]` is a DTP of type *member* that is a checked prompt.
- `[String1]` is a DTP of type *string* that is a checked prompt.

Assume that the following values are input:

- [Mbr1]: "Joe Smith"
- [String1]: "Smith"

In this example, [DTP_Start] returns "false."

@SubString

@SUBSTRING returns the substring that begins with the character at the beginning index and extends to the character before the ending index. The beginning index starts with zero, and the text that is returned does not include the ending index. If the ending index is greater than the number of characters in the string, then nothing is returned.

DTP Type:

@SUBSTRING can be assigned to a DTP of type *member*, *members*, *dimension*, *dimensions*, *string*, or *password*.

Syntax:

@SUBSTRING(*String,Beginning Index,Ending Index*)

Parameters:

Parameter	Description
String	Design-time prompt or text
Beginning Index	Design-time prompt *numeric* or *text*
Ending Index	Design-time prompt of the *numeric* or *text*

Example:

[SUBSTRING_Str]=@SUBSTRING([String1],[start_ind],[end_ind])

Where:

- [SUBSTRING_Str] is a DTP of type *string* that is not a checked prompt.
- [String1] is a DTP of type *string* that is a checked prompt.
- [start_ind] is a DTP of type *numeric* that is a checked prompt.
- [end_ind] is a DTP of type *numeric* that is a checked prompt.

Assume that the parameters are as follows:

- String: "This is a test of converting characters to strings."
- Beginning Index: "2"
- Ending Index: "16"

In this example, [SUBSTRING_Str] returns "his is a test o" because the second character in the string is the "h" in "this" and the 15th character is the "o" at the beginning of the word "of".

@ToLowerCase

@TOLOWERCASE changes the characters in a string to lower case.

DTP Type:

`@TOLOWERCASE` can be assigned to a DTP of type *member*, *members*, *dimension*, *dimensions*, *string*, or *password*.

Syntax:

```
@TOLOWERCASE(String)
```

Parameter:

Parameter	Description
String	Design-time prompt or text

Example:

```
[TOLOWERCASE_Str]=@TOLOWERCASE([String1])
```

Where:

- `[TOLOWERCASE_Str]` is a DTP of type *string* that is not a checked prompt.
- `[String1]` is a DTP of type *string* that is a checked prompt.

Assume that the parameter is as follows:

String: "Testing for Today"

In this example, `[TOLOWERCASE_Str]` returns "Testing for Today"

@ToMDX

`@TOMDX` returns an MDX expression.

DTP Type:

`@TOMDX` can be assigned to a DTP of type *string*.

Syntax:

```
@TOMDX(Members, Filter Shared, Non Empty, Generate Crossjoin)
```

Parameters:

Parameter	Description
Members	Design-time prompt, member, or a function (design-time prompt or function returns a member or members)
Filter Shared	**Optional.** True or False. True adds the MDX syntax to filter shared members. The default is False.
Non Empty	**Optional.** True or False. True adds `NON EMPTY` in front of the MDX syntax. The default is False.
Generate Crossjoin	**Optional.** True or False. True returns the MDX syntax with the crossjoin.

Examples:

```
[MDX_function1]=@TOMDX(@UNION([MR],[MR2]),"true","true","true")
```

```
[MDX_function2]=@TOMDX(@UNION([MR],[MR2]),"false","false","false")
```

Where:

- `[MDX_function]` is a non-promptable DTP *string*.
- `[MR1]` is a promptable DTP *member*.
- `[MR2]` is a promptable DTP *member range*.

Assume that `[MR1]` has the following inputs:

- Account: @Relative("Gross Margin, %, 0")
- Period: "Apr", "May"
- HSP_View: "BaseData"
- Year: "FY15"
- Scenario: "Plan", "Actual"
- Version: "Working"
- Entity: @Relative("100", 0)
- Product: "P_110", "P_150", "P-100"

Assume that `[MR2]` has the following inputs:

- Account:
- Period: "Jan, Feb, Mar"
- HSP_View:
- Year:
- Scenario:
- Version: "Final"
- Entity:
- Product:

In this example:

`[MDX_function1]` returns:

```
NON EMPTY
(Crossjoin(Crossjoin(Crossjoin(Crossjoin(Crossjoin(Crossjoin(Crossjoin(FILTER
({RELATIVE([Gross Margin %],  0}, NOT
[Account].CurrentMember.SHARED_FLAG),FILTER ({[Apr],[May],Jan,Feb,Mar}, NOT
[Period].CurrentMember.SHARED_FLAG)),FILTER ({[BaseData]}, NOT
[HSP_View].CurrentMember.SHARED_FLAG)),FILTER ({[FY15]}, NOT
[Year].CurrentMember.SHARED_FLAG)),FILTER ({[Plan],[Actual]}, NOT
[Scenario].CurrentMember.SHARED_FLAG)),FILTER ({[Working],[Final]}, NOT
[Version].CurrentMember.SHARED_FLAG)),FILTER ({[110]}, NOT
[Entity].CurrentMember.SHARED_FLAG)),FILTER ({[P_110],[P_150],[P_000]}, NOT
[Product].CurrentMember.SHARED_FLAG)))
```

`[MDX_function2]` returns:

```
[@Relative("Gross Margin %", 0)],[Apr","May",Jan,Feb,Mar],[BaseData],[FY15],
[Plan","Actual],[Working","Final],[110],[P_110","P_150","P_000]
```

@ToUpperCase

`@TOUPPERCASE` changes the characters in the string to upper case.

DTP Type:

`@TOUPPERCASE` can be assigned to a DTP of type *member*, *members*, *dimension*, *dimensions*, *string*, or *password*.

Syntax:

```
@TOUPPERCASE(String)
```

Parameter:

Parameter	Description
String	Design-time prompt or text

Example:

```
[TOUPPERCASE_Str]=@TOUPPERCASE([String1])
```

Where:

- `[TOUPPERCASE_Str]` is a DTP of type *string* that is not a checked prompt.
- `[String1]` is a DTP of type *string* that is a checked prompt.

Assume that the parameter is as follows:

String: "Testing to convert to upper case."

In this example, `[TOUPPERCASE_Str]` returns "TESTING TO CONVERT TO UPPER CASE."

@Trim

`@TRIM` removes leading or trailing white spaces.

DTP Type:

`@TRIM` can be assigned to a DTP of type *member*, *members*, *string*, or *password*.

Syntax:

```
@TRIM(String)
```

Parameter:

Parameter	Description
String	Design-time prompt or text

Example:

```
[TRIM_ST]=@TRIM([String1])
```

Where:

- `[TRIM_ST]` is a DTP of type *string* that is not a checked prompt.
- `[String1]` is a DTP of type *string* that is a checked prompt.

Assume that the parameter is as follows:

String: "This is a test of leading and trailing spaces"

The parameter has white spaces before and after the text. It displays as:

```
String1 =       this is a test of leading and trailing spaces
```

@Union

`@UNION` returns the combination of members used in multiple design-time prompts.

DTP Type:

`@UNION` can be assigned to a DTP of type *member range* or *cross dimension*.

Syntax:

`@UNION(Argument1,Argument2)`

Parameters:

Parameter	Description
Argument1	Design-time prompt, function, or members
Argument2	Design-time prompt, function, or members

Example:

`[UnionMR]=@UNION([MR1],[MR2])`

Where:

- `[UnionMR]` is a DTP of type *member range* that is not a checked prompt.
- `[MR1]` is a DTP of type *member range* that is a checked prompt.
- `[MR2]` is a DTP of type *member range* that is a checked prompt.

Assume that the input for `MR1` and `MR2` is:

- `[MR1]`: "50350",@Relative("Q1", 0),"California","Tennessee"
- `[MR2]`: "Apr","Washington", "Tennessee"

In this example, `UnionMR` returns:

"50350",@Relative ("Q1", 0),"Apr","California","Tennessee","Washington"

@ValueDimCount

`@VALUEDIMCOUNT` returns the number of dimensions for which a member was specified.

DTP Type:

`@VALUEDIMCOUNT` can be assigned to a DTP of type *numeric*.

Syntax:

`(@VALUEDIMCOUNT(DTP,Dense/Sparse)`

Parameters:

Parameter	Description
DTP	Design-time prompt or function that returns a member range or cross dimension.
`Dense/Sparse`	**Optional.** If left empty, returns the count of all dimensions that have a member specified. If "Dense" or "Sparse" is entered, only the specified type of dimensions that have a member specified are counted.

Example 1:

```
[DTP1]=@VALUEDIMCOUNT([MR1])
```

Where:

- `[DTP1]` is a DTP of type *numeric* and is not a checked prompt.
- `[MR1]` is a DTP of type *member range* with the following inputs:
  - Account:
  - Period: Jan, Feb, Mar
  - HSP_View:
  - Year:
  - Scenario:
  - Version: Working
  - Entity: Sparse
  - Product: P-000

In this example, `[DTP1]` returns "3", since there are three dimensions that have members specified.

Example 2:

```
[DTP2]=@VALUEDIMCOUNT([MR1],"Dense")
```

Where:

- `[DTP2]` is a DTP of type *numeric* and is not a checked prompt.
- `[MR1]` is a DTP of type *member range* with the following inputs:
  - Account: Dense
  - Period: Dense
  - HSP_View: Sparse
  - Year: Sparse
  - Scenario: Sparse
  - Version: Sparse
  - Entity: Sparse

– Product: Sparse

In this example, `[DTP2]` returns "1", since "Account" and "Period" are dense, and "Period" has members specified.

Sharing Script and Formula Components

A shared formula or script component exists at the formula level and script level.

Related Topics

- About Sharing Script and Formula Components
  You can share formula and script components across Oracle Hyperion Financial Management, Planning, and Oracle Essbase block storage business rules and templates that belong to same application type.
- Changing Formula and Script Components from Shared to Not Shared
- Changing Formula and Script Components from Not Shared to Shared

About Sharing Script and Formula Components

You can share formula and script components across Oracle Hyperion Financial Management, Planning, and Oracle Essbase block storage business rules and templates that belong to same application type.

You can share formula and script components across Planning business rules and templates.

> **Note**
>
> You cannot share member range, condition, fixed loop, member block, or DTP Assignment components.

A shared formula or script component exists at the formula level and script level. Shared formula and script components are used so that when you change a formula or script, the change is reflected in all rules and templates in which the shared component is used. When you share a component, the system creates a cross reference to the original component. By creating a cross reference to, instead of a copy of, the original component, less space is used in the database and processing time may be decreased.

Changing Formula and Script Components from Shared to Not Shared

Before you change a shared formula or script component to not shared, you must ensure that it is not used in more than one business rule or template. You can use the Show Usages feature to see which business rules and templates use the formula or script component. (See Showing the Usages of Formula and Script Components.) Then you can create copies of the shared component for each business rule and template in which it is used by clearing the Shared check box for the component from within the rules and templates.

To change a formula or script component from shared to not shared:

1. In **System View**, right-click the rule or template, and then select **Open**.
2. In the **Rule Designer** or **Template Designer**, do one of these tasks:

- If you do not want to share a formula or script component you are adding to a flow chart, after you drag the formula or script component into the flow chart, clear **Shared** on **Properties**.
- If you do not want to share a formula or script component that is already in the flow chart, select the component in the flow chart, and clear **Shared** on **Properties**.

This creates a copy of the component in the rule or template.

3. Click  .

Changing Formula and Script Components from Not Shared to Shared

To change a formula or script component from not shared to shared, you must ensure there is not another shared or unshared formula or script component that has the same name *within the Planning application type*. Shared objects must have unique names across applications, so you cannot create a shared object with a name that is already used.

To share a formula or script component, ensure that the Shared check box is selected on the component's Properties tab when you drag and drop an existing component into a rule or template's flow chart. (The Shared check box is selected by default.)

To change a formula or script component from not shared to shared:

1. In **System View**, right-click the rule or template, and then select **Open**.
2. When the **Rule Designer** or **Template Designer** opens, do one of these tasks:
   - To share a formula or script component:
     - In **New Objects**, drag the formula or script component, and drop it into the desired location in the flow chart.
     - On the component's **Properties** tab, select **Shared**, and name the component.
   - To share a formula or script component that is already in the flow chart:
     a. Select the component in the flow chart.
     b. On the component's **Properties** tab, select **Shared**, and name the component. All shared components must have a name.
3. Select **, Save**.

Copying Components

Copy and paste the children of a business rule component, or copy and paste the reference to a business rule formula or script component.

Related Topics

- Copying and Pasting the Children of a Business Rule Component
- Copying and Pasting the Reference to a Business Rule Formula or Script Component

Copying and Pasting the Children of a Business Rule Component

When you are working with a business rule that has components, you may want to copy a component's children (that is, the components that are grouped underneath a component in a flow chart). You can copy the children of components and paste them into the same business rule or into a different business rule.

To copy and paste the children of a business rule component:

1. Open the business rule.
2. In the business rule flow chart, right-click the member range, fixed loop, or condition component whose children you want to copy, and then select **Copy Group**.
3. Do one of these tasks:
   - To paste the component's children into the *same* business rule, right-click the component to the left of the location where you want the component's children to display in the flow chart, and then select **Paste**. (The component's children display to the right of the component you select.)
   - To paste the component's children into a *different* business rule, open the business rule into which you want to paste the component's children, and right-click the component to the left of the location where you want the component's children to display in the flow chart, and then select **Paste**.
4. Click  .

> **Note**
>
> When you copy and paste the children of a component, any shared components are also copied.

Copying and Pasting the Reference to a Business Rule Formula or Script Component

When you copy and paste a reference to a business rule shared script or shared formula component, you copy and paste a shortcut to the component; the component itself is not copied. Only one copy of the component exists in the original business rule from which you copied the reference. The reference functions as a pointer to the application and plan type that contains the shared component. If you select copy reference on a component that is not shared, the pasted component is a copy of the component, and has no shortcut to the original component. In this instance, copy reference behaves the same as copy.

You can copy and paste a reference to a shared formula or shared script component within the same business rule or within a different business rule.

To copy and paste a reference to a shared formula or a shared script component:

1. Right-click the rule that contains the shared formula or shared script component, and then select **Open**.
2. In the Rule Designer flow chart, right-click the shared formula or shared script component, and then select **Copy Reference**.
3. Do one of these tasks:
   - To paste the component's reference into the *same* business rule, right-click the component to the left of the location you want the component's reference to display in the flow chart. (The component's reference displays to the right of the component you select.) Then select **Paste**.
   - To paste the component's reference into a *different* business rule, open the business rule into which you want to paste the component's reference, and right-click the

component to the left of the location you want the component's reference to display in the flow chart. Then click **Paste**.

> **Note**
>
> You can copy and paste the reference into a business rule that belongs to the same application or to a different application, as long as the application belongs to the same application type.

4. Click  .

Saving Components

You save formula and script components after you design them in the Component Designer.

Related Topics

- Saving a Component
- Saving Formula and Script Components with a Different Name

Saving a Component

You save formula and script components after you design them in the Component Designer. You save condition, range, and fixed loop components when you save the business rule or template to which they belong in the Rule Designer or Template Designer.

If you are working with a component, you can save it as a custom template. If you are working with a formula or script component, you can save it with a different name.

To save a component, after you are finished designing it in the Component Designer, the Template Designer, or the Rule Designer, click  .

Saving Formula and Script Components with a Different Name

You can save script and formula components with a different name using Save As. Save As creates a copy of the formula or script component. You may want to create a copy of a component if it is a shared component, and you do not want it to be shared. See Sharing Script and Formula Components.

To save a script or formula component with a different name:

1. In **System View**, right-click a formula or script, and then select **Open**.
2. In the Component Designer, select **File**, and then **Save As**.
3. In **Save As**, enter the formula or script's new name, and select an application. Then select a **Plan Type**.
4. Click **OK**.

   After you save it, you may need to refresh the application list in the System View to see the formula or script component. See Refreshing Formula and Script Components.

Refreshing Formula and Script Components

After you create a formula or script component, you may need to refresh the application list in the System View to see it in the Formulas or Scripts node.

When you refresh the application type, application, or calculation or plan type to which a formula or script component belongs, you refresh the formula and script components by default. Refreshing formula or script components, however, does not refresh higher levels (that is, calculation or plan types, applications, or application types) in the application list.

To refresh formula or script components, un **System View**, right-click **Scripts** or **Formulas**, and then select **Refresh**.

> **Note**
>
> You can also refresh higher levels in the database outline to refresh the objects within them. For example, to refresh scripts and formulas for an application, right-click the application name, and then select **Refresh**.

Showing the Usages of Formula and Script Components

You can show the usages of script or formula components.

To show the usages of a script or formula component:

1. In **System View**, right-click a script or a formula, and then select **Show Usages**.
2. Review the information in the **Usages** window, and then click **OK**.

Working with Components in a Flow Chart

You can perform actions on the components in a flow chart.

Related Topics

- About Working With Components in a Flow Chart
- Collapsing and Expanding a Component in a Flow Chart
- Removing a Component from a Flow Chart
- Copying and Pasting a Component in a Flow Chart
- Copying and Pasting a Reference to a Component in a Flow Chart
- Copying and Pasting a Component Group in a Flow Chart

About Working With Components in a Flow Chart

You can perform actions on the components in a flow chart, including expanding and collapsing components to show or hide detail, removing components, saving components as a template, copying and pasting components, and copying and pasting component groups and component references. You can also undo many of the changes you make to components while working with them in the Template Designer and Rule Designer flow charts.

Collapsing and Expanding a Component in a Flow Chart

If you have a business rule or template that has many complex components, you may want to collapse or expand some of them in the flow chart. By collapsing and expanding components in a flow chart, you can maximize space for the display of components you want to work with while minimizing space for the display of those with which you do not want to work.

To collapse a component in a flow chart:

1. In **System View**, right-click a business rule or template, and then select **Open**.
2. In the **Rule Designer** or **Template Designer** flow chart, perform one of these actions on the components:
   - To expand a component, right-click it, and select **Expand**.
   - To collapse a component, right-click it, and then select **Collapse**.
3. Click  .

Removing a Component from a Flow Chart

Removing a condition, member range, data range, or fixed loop component from a business rule or template's flow chart deletes the component. These components cannot be shared, so they exist in only one business rule or template.

Removing formula or script components from a flow chart may or may not delete the component. If the formula or script component is *not shared*, when you remove the component, it is deleted. If the formula or script component *is shared*, it is only deleted from the business rule or template from which you remove it.

To remove a component from a flow chart:

1. In **System View**, right-click the business rule or template, and then select **Open**.
2. In the flow chart of the **Rule Designer** or **Template Designer**, right-click the component you want to remove, and then select **Remove**.
3. Click  .

Copying and Pasting a Component in a Flow Chart

When you a copy a component in a business rule or template's flow chart, you can paste it into a different location within the same business rule or template's flow chart, or paste it into the flow chart of a different business rule or template, if the business rule or template belongs to the same application type.

To copy and paste a component:

1. In **System View**, right-click the business rule or template, and then select **Open**.
2. In the flow chart of the **Rule Designer** or **Template Designer**, right-click the component you want to copy, and select **Copy**.

> **Tip**
>
> You can also use Edit, Copy.

3. Do one of these tasks:
   - To paste the component into the *same* business rule or template, right-click the location in the flow chart where you want to paste the component, and select **Paste**.
   - To paste the component into a *different* business rule or template, open the business rule or template, right-click the location in the flow chart where you want to paste the component, and select **Paste**.
4. Click  .

Copying and Pasting a Reference to a Component in a Flow Chart

Unlike copying a component itself, copying a reference to a component copies only the pointer to the component. After you copy the reference to the component, the component itself exists only in the original location in which you created it.

When you copy a reference to a component, you can paste it into the same business rule or template, or you can paste it into a different business rule or template that belongs to the same application type.

To copy and paste a reference to a component in a flow chart:

1. In **System View**, right-click the business rule or template, and select **Open**.
2. In the **Rule Designer** or **Template Designer**, right-click the component whose reference you want to copy in the flow chart, and select **Copy Reference**.

> **Tip**
>
> You can also use Edit, Copy Reference.

> **Note**
>
> Copy Reference is only available for shared components (that is, formulas and script) or rules and templates that are used in the flow of another rule or template.

3. Do one of these tasks:
   - To paste the component's reference into the *same* business rule or template, right-click the location in the flow chart where you want to paste the reference, and select **Paste**.
   - To paste the component's reference into a *different* business rule or template, open the business rule or template, and right-click the location in the flow chart where you want to paste the reference, and select **Paste**.
4. Click  .

Copying and Pasting a Component Group in a Flow Chart

If a component contains other components (that is, if there are components within the component), you can copy the group of components and paste it into another location within the same flow chart, or paste it into another flow chart.

To copy and paste a component group in a flow chart:

1. In **System View**, right-click the business rule or template, and select **Open**.
2. In the **Rule Designer** or **Template Designer**, right-click the component group you want to copy in the flow chart, and select **Copy Group**.

> **Tip**
>
> You can also use Edit, Copy Group.

3. Do one of these tasks:
   - To paste the component group into the *same* business rule or template, right-click in the flow chart where you want to paste the group, and select **Paste**.
   - To paste the component group into a *different* business rule or template, open the business rule or template, right-click in the flow chart where you want to paste the group, and select **Paste**.
4. Click  .

8

Using Aggregate Storage Components to Design Business Rules

Related Topics

- About Using Aggregate Storage Components to Design Business Rules
  Use aggregate storage components to design business rules.
- Working with Point of View Components
  You can create and edit point of view components.
- Working with Allocation Components
  You can create and edit allocation components.
- Opening a Point of View or Allocation Component
  You open a point of view or allocation component from within the flow chart of the business rule to which it belongs; you cannot open it from the System View.
- Deleting a Point of View or Allocation Component
  You delete a point of view or allocation component by removing it from the business rule to which it belongs.
- Copying and Pasting a Point of View or Allocation Component
  You can copy a point of view or allocation component from a business rule, and then you can paste the component into the same or a different business rule.
- Saving a Point of View or Allocation Component
  You save point of view or allocation component when you save the business rule to which it belongs in the Rule Designer.
- Working with Aggregate Storage Formula Components
  Create, open, edit, delete, copy, paste, and show usages for aggregate storage formula components.

About Using Aggregate Storage Components to Design Business Rules

Use aggregate storage components to design business rules.

Business rules in Planning aggregate storage applications are composed of different components than business rules in Planning block storage applications. :

These are the three components you use to design business rules in Planning aggregate storage applications

- Point of View components contain lists of metadata members (for example, lists of accounts).

> **Note**
>
> You can nest a Point of View component within another Point of View component.

- Allocation components contain calculations for distributing data from members at one level in the database outline to other members in the outline.
- Formula components contain calculation statements that you design using members, functions, and variables.

As you create components, you may want to leave the business rules, components, templates and variables you're working with open. Calculation Manager displays these objects in a tabbed interface so you can move easily among the tabs as you are creating components. You can have as many as ten tabs open within Calculation Manager, but Oracle recommends that you not open more than ten objects simultaneously for optimum performance.

Working with Point of View Components

You can create and edit point of view components.

Related Topics

- Creating a Point of View Component
- Editing a Point of View Component

Creating a Point of View Component

You create Point of View component from within a business rule.

To create a Point of View component:

1. In **System View**, right-click a rule, and then select **Open**.

   The business rule is displayed in the Rule Designer.
2. After you determine where in the business rule's flow chart you want to create the Point of View component, from the **New Objects Palette**, drag the **Point of View** component and drop it into that location in the flow chart.

   The Point of View object is displayed as two circles with arrows inside them.
3. On **Point of View**, enter a caption to identify the Point of View component. The caption is displayed above the component in the flow chart of any rule that uses it.
4. **Optional:** Do one of these tasks to define the Point of View's global range:

   > **Note**
   >
   > If a global range is defined for the business rule for which you are creating the Point of View component, the Point of View tab displays the business rule's member selections by default. To see if a global range is defined for the business rule, select the Begin or End tab in the flow chart. Then click on the Global Range tab to see if any members, functions, or variables are defined.

   - Click **Variable Selector** to select or create variables to define the point of view. If you select a variable, you can select **Link Variable Dynamically** to ensure the variable is updated dynamically when changes are made to it.
   - Click **Member Selector** to select members to define the point of view.
   - Click in the row of a dimension in the **Value** column to type the names of members that define the point of view.

- Click in the row of a dimension, then click the **Actions** icon, and then select one of these options to enter members:
  - Members (see Adding Members and Functions to a Component)
  - Variables (see Working with Variables)

    For non-groovy rules, variables of type *member* or *members* are the only supported types of variables for Planning cubes of type *Aggregate Storage Option* (ASO)
  - Functions (see Working with Functions)

    The functions you enter should return level 0 members only and should include an @ symbol before the function name. For any function that has a `List` parameter, the input must be a single member or a function that returns a member or member list.

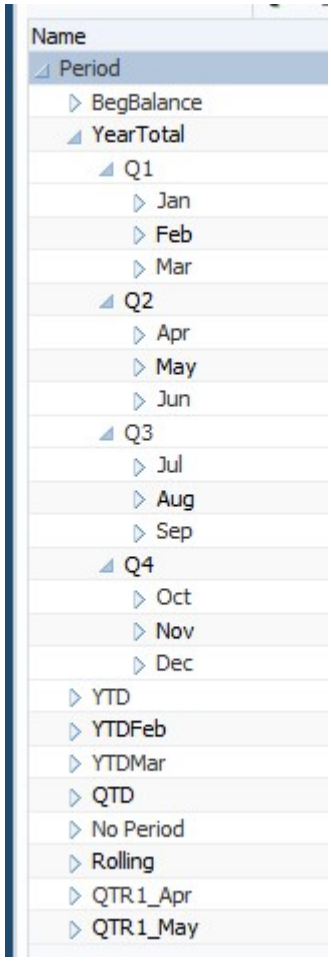
    You can enter these functions:

    * `@ANCESTOR(Member Name,Index|Layer)` given the input member, returns an ancestor at the specified index or layer. (Use `@ANCESTOR` only in the rule's POV.)
    * `@ATTRIBUTE(Attribute Member Name)` generates a list of all base members that are associated with the specified attribute member.
    * `@COUSIN(Member Name,Cousin Member Name)` returns a child member at the same position as a member from another ancestor. (Use `@COUSIN` only in the rule's POV.)
    * `@DISTINCT(List1)` deletes duplicate tuples from a set. (Use `@DISTINCT` only in the rule's POV.)
    * `@EXCEPT(List1,List2)` returns a subset containing the differences between two sets, optionally retaining duplicates. (Use `@EXCEPT` only in the rule's POV.)
    * `@FilterDynamic(Dimension Name,Member Name)` removes all dynamic members from the list of members. (Use `@FilterDynamic` only in the rule's POV.)
    * `@FilterShared(Dimension Name,Member Name)` removes all shared members from the list of members. (Use `@FilterShared` only in the rule's POV.)
    * `@FilterSharedAndDynamic(Dimension Name,Member Name)` removes all dynamic and shared members from the list of members. (Use `@FilterSharedAndDynamic` only in the rule's POV.)
    * `@FIRSTCHILD(Member Name)` returns the first child of the input member. (Use `@FIRSTCHILD` only in the rule's POV.)
    * `@FIRSTSIBLING(Member Name)` returns the first child of the input member's parent. Use `@FIRSTSIBLING` only in the global POV.
    * `@GEN(Member Name)` returns the members specified by the input generation number of the specified member. (Use `@GEN` only in the rule's POV.)
    * `@GENMBRS(Member Name,Generation)` returns the generation specified by the input generation number. (Use `@GENMBRS` only in the rule's POV.)
    * `@ILSIBLINGS` returns the specified member and its left siblings. (Use `@ILSIBLINGS` only in the rule's POV.)
    * `@INTERSECT(List1,List2)` returns the intersection of two input sets, optionally retaining duplicates. (Use `@INTERSECT` only in the rule's POV.)

* `@IRSIBLINGS` returns the specified member and its right siblings. (Use `@IRSIBLINGS` only in the rule's POV.)
* `@LAGGEN(Member Name,Index)` using the order of members existing in a database outline, returns a member that is n steps behind a given member, along the same generation. Use `@LAGGEN` only in the global POV.
* `@LAGLEV(Member Name,Index)` using the order of members existing in a database outline, returns a member that is n steps behind a given member, along the same level. Use `@LAGLEV` only in the global POV.
* `@LASTCHILD(Member Name)` returns the last child of the input member. (Use `@LASTCHILD` only in the rule's POV.)
* `@LASTSIBLING(Member Name)` returns the last child of the input member's parent. Use `@LASTSIBLING` only in the global POV.
* `@LAYERGEN(Member Name,Index)` returns the generation based layer for the member specified.

  `@LAYERGEN` can only be used within a function where the parameter is looking for layer; for example, `@ANCESTOR(member,layer,index)`. You cannot use `@LAYERGEN` by itself. Use `@LAYERGEN` only in the rule's POV.

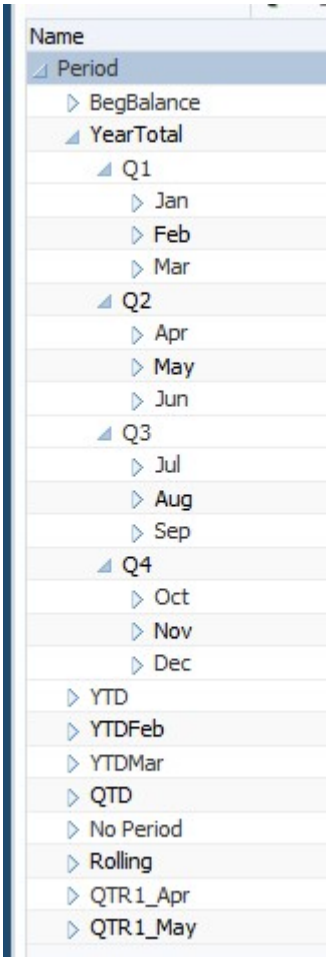
  In the following Period dimension, `@Level0Descendants(@ANCESTOR(Jul,@LAYERGEN(Period,3)))` returns Jul, Aug, Sep



* `@LAYERLEVEL(Member Name,Index)` returns the level based layer for the member specified.

  `@LAYERLEVEL` can only be used within a function where the parameter is looking for layer; for example, `@ANCESTOR(member,layer,index)`. You cannot use `@LAYERLEVEL` by itself. Use `@LAYERLEVEL` only in the rule's POV.

  In the following Period dimension, `@ANCESTOR(Oct,@LAYERLEVEL(Period,0))` returns the members Oct, Nov, Dec



* `@LEADGEN(Member Name,Index)` using the order of members existing in a database outline, returns a member that is *n* steps past a given member, along the same generation. Use `@LEADGEN` only in the global POV.
* `@LEADLEV(Member Name,Index)` using the order of members existing in a database outline, returns a member that is n steps past a given member, along the same level. Use `@LEADLEV` only in the global POV.
* `@Level0Descendant(Member Name)` expands to include all level zero descendants of the specified member.
* `@LIST(Argument)` returns a list of members. (Use `@LIST` only in the rule's POV.)

* `@LSIBLINGS` returns the left siblings of the specified member. (Use `@LSIBLINGS` only in the rule's POV.)
* `@NEXTLEVMBR(Member Name)` using the order of members existing in a database outline, returns the next member along the same level. Use `@NEXTLEVMBR` only in the global POV.
* `@NEXTMBR(Member Name)` using the order of members existing in a database outline, returns the next member along the same generation or level. (Use `@NEXTMBR` only in the rule's POV.)
* `@PARENT(Member Name)` returns a member's parent. (Use `@PARENT` only in the rule's POV.)
* `@PREVLEVMBR(Member Name)` using the order of members existing in a database outline, returns the previous member along the same level. Use `@PREVLEVMBR` only in the global POV.
* `@PREVMBR(Member Name)` using the order of members existing in a database outline, returns the previous member along the same generation or level. (Use `@PREVMBR` only in the rule's POV.)
* `@RSIBLINGS` returns the right siblings of the specified member. (Use `@RSIBLINGS` only in the rule's POV.)
* `@Siblings(Member Name)` expands to include all siblings of the specified member.
* `@UDA(Dimension Name, User-Defined Attribute String)` selects members based on a common attribute, which you define as a user-defined attribute (UDA) on the Oracle Essbase server.

5. To enter a comment for the members you select for a dimension, click **Comment**.
6. Click **Reset Grid** to clear any entries you made to the grid.
7. Click  .

Editing a Point of View Component

You can edit the members, variables, and functions that comprise the global range of the Point of View component. You can also edit the caption that displays above the component in a flow chart and the comments that are entered for the values selected for each of the dimensions in the Point of View.

To edit a Point of View component:

1. In **System View**, right-click the business rule that contains the Point of View, and then select **Open**.
2. In the Rule Designer, select the Point of View component you want to edit in the flow chart to display its properties. You can edit any of these properties of a Point of View component. (See Creating a Point of View Component.)
   - The caption that displays above the Point of View component in the business rule's flow chart
   - The members, variables, and functions that define the Point of View
   - Whether any variables used in the Point of View component are updated dynamically when changes are made to the variables

- Whether comments are entered for the dimensions and members that define the global range of the Point of View
- Whether the values of the members in the Point of View component are calculated when the business rule to which it belongs is validated or launched

3. Click  .

Working with Allocation Components

You can create and edit allocation components.

Related Topics

- Creating an Allocation Component
- Editing an Allocation Component

Creating an Allocation Component

An allocation component enables you to distribute data from a member to level 0 descendents of that member. You create an allocation component from within a business rule; it exists only in that business rule and cannot be shared amongst business rules.

To create an allocation component:

1. In **System View**, right-click a rule, and then select **Open**.

   The business rule is displayed in the Rule Designer.

2. After you determine where in the business rule's flow chart you want to create the allocation component, from the **New Objects Palette**, drag the **Allocation** component and drop it into that location in the flow chart.

   > **Note**
   >
   > If you drop a point of view component within another point of view component, the second point of view inherits the members, variables, and functions from the first (that is, upper) point of view.

3. In the Allocate Wizard, in **Point of View**, for each dimension listed that you do not want to vary during the allocation, do one of these tasks:
   - Select a predefined selection from **Use Predefined Selection** to populate the dimensions listed with values.
   - Click the **Member Selector** icon to select members and variables for each of the dimensions listed.

> **Note**
>
> In the Member Selector, the dimensions listed in the current step of the wizard are available for selection from Dimension. This enables you to select members and functions for any of the dimensions listed in the current step of the wizard.
>
> Make sure that all members you select are valid level 0 members.

- Select a dimension in the list, and then click **Actions** to select a member or variable. See Using Member Selection, Variables, Functions, Smart Lists, and Planning Formula Expressions to Design Components.

4. In the Allocate Wizard, in **Source**, for each dimension listed, select a member whose data you want to allocate by doing one of these tasks.

> **Note**
>
> You must select a member for each dimension listed.
>
> The source members can be non-level 0 members.

- Select a predefined selection from **Use Predefined Selection** to populate the dimensions listed with values.

> **Note**
>
> If the predefined selection does not enter a value for each dimension listed, you must enter a value for any dimensions that are empty.

- Click the **Member Selector** icon to select a member for each of the dimensions listed.
- Select a dimension in the list, and click **Actions** to select a member or variable. See Using Member Selection, Variables, Functions, Smart Lists, and Planning Formula Expressions to Design Components.

> **Note**
>
> You cannot use functions in this step of the Allocation component.

- In **Optional**, to allocate a specific value, enter an amount to be allocated instead of the selections above.

5. If the source amount to allocate is zero, select one of these options from the drop-down.
   - Select the next pool record.
   - Stop processing the allocation.
6. In **Allocation Range**, do the following:
   - Enter the parent member for the dimensions to use for the allocation.

     To enter the parent member, do one of these tasks:

- Select a predefined selection from **Use Predefined Selection** to populate the dimensions listed with values.
- Click the **Member Selector** icon to select the parent member for the dimension to which to allocate the data.
- Enter a parent member, or select a dimension in the list, and click the **Actions** icon to select the parent member (of the main dimension) to which to allocate the data. See Using Member Selection, Variables, Functions, Smart Lists, and Planning Formula Expressions to Design Components.

  The data is allocated to the level 0 member (that is, the lowest member in the outline, with no members beneath it) below the parent member in the database outline.

- Select whether to clear the region before the allocation process.

  If you select **Yes** to clear the region, you will enter the members to be cleared before the allocation is run in the **Clear Region** field (see Step 8).

7. In **Target**, for the remaining dimensions, select a level 0 member to which to allocate the data. Do one of these tasks:
   - Select a predefined selection from **Use Predefined Selection** to populate the dimensions listed with values.
   - Click the **Member Selector** icon to select members for each of the dimensions listed.
   - Select a dimension in the list, and click the **Actions** icon to select a member or variable. See Using Member Selection, Variables, Functions, Smart Lists, and Planning Formula Expressions to Design Components.
8. In **Clear Region**, enter the level 0 member(s) to be cleared with a logical clear before the allocation process. Do one of these tasks:
   - Select a predefined selection from **Use Predefined Selection** to populate the dimensions listed with values.
   - Click the **Member Selector** icon to select members for each of the dimensions listed.
   - Select a dimension in the list, and click the **Actions** icon to select a member or variable. See Using Member Selection, Variables, Functions, Smart Lists, and Planning Formula Expressions to Design Components.

   > **Note**
   >
   > If you do not see **Clear Region**, you did not select **Yes** in the **Allocation Range** to clear the data before allocation.

9. In **Offset**, do one of these tasks:
   - Select a predefined selection from **Use Predefined Selection** to populate the dimensions listed with values.
   - Click the **Member Selector** icon to select members for each of the dimensions listed.
   - Select a dimension in the list, and click the **Actions** icon to select a member or variable. See Using Member Selection, Variables, Functions, Smart Lists, and Planning Formula Expressions to Design Components.

> **Note**
>
> You must specify members for the offset; you cannot leave it empty.

10. **Optional:** In **Exclude**, select any members you want to exclude from the allocation. Do one of these tasks:
    - Select a predefined selection from **Use Predefined Selection** to populate the dimensions listed with values.
    - Click the **Member Selector** icon to select members for each of the dimensions listed.
    - Select a dimension in the list, and click the **Actions** icon to select a member or variable. See Using Member Selection, Variables, Functions, Smart Lists, and Planning Formula Expressions to Design Components.
11. In **Basis**, do these tasks:
    a. Select an allocation method to specify how the data should be allocated.
       - Select **Allocate evenly** to allocate data values in the allocation range evenly. Then in **Basis Options for evenly method**, specify what you want to be done if the basis is negative, zero, has missing values, or if all members are excluded.
       - Select **Allocate using a driver** to calculate a percentage to be applied to each member in the allocation range. Then in **Basis Options**, specify what you want to be done if the basis is negative or equal to zero.

    b. Any dimension members you do not specify are inherited from the POV you defined previously, but you can override those POV selections by doing one of these tasks:
       - Select a predefined selection from **Use Predefined Selection** to populate the dimensions listed with values.
       - Click the **Member Selector** icon to select a member for each of the dimensions listed.
       - Select a dimension in the list, and click the **Actions** icon to select a member or variable. See Using Member Selection, Variables, Functions, Smart Lists, and Planning Formula Expressions to Design Components.

    c. Click **Next**.
12. In **Rounding**, complete these steps:
    a. Enter the number of decimal places to use for this allocation, or click the **Actions** icon to select a member or variable that represents this value.
    b. Select where to place the rounding difference.
       - Select **Define location** to specify a member or members on which to place the rounding difference.
       - Select **Use biggest value** to round data values to their largest values
       - Select **Use smallest value** to round data values to their smallest values.
       - Select **Discard rounding error** to use allocated data values as they are.
13. If you selected **Define location** in the previous step, in **Rounding member**, do one of these tasks:
    - Select a predefined selection from **Use Predefined Selection** to populate the dimensions listed with values.
    - Click the **Member Selector** icon to select a member for each of the dimensions listed.

- Select a dimension in the list, and click the **Actions** icon to select a member or variable. See Using Member Selection, Variables, Functions, Smart Lists, and Planning Formula Expressions to Design Components.

> **Note**
>
> The members you select in this step must be a part of the allocation range.

14. Click **Finish**.

Editing an Allocation Component

You can edit an allocation component by opening the business rule to which it belongs. When the business rule is displayed in the Rule Designer, you can view the allocation component's properties by selecting it in the business rule's flow chart.

To edit an allocation component:

1. In **System View**, right-click the business rule that contains the allocation component, and then select **Open**.
2. In the Rule Designer, select the allocation component you want to edit in the flow chart to display its properties. You can edit any of these properties of an allocation component. (See Creating an Allocation Component.)
   - The member whose data you want to allocate
   - The level 0 members to which you want to allocate data
   - The data and the amount of the data you want to allocate
   - Whether you want the total amount of the data allocated to be written to an offset member
   - Whether you want the data to be allocated evenly or to be allocated in different amounts using a driver
   - Whether the allocated data should be rounded, and if so, how it should be rounded
3. Click  .

Opening a Point of View or Allocation Component

You open a point of view or allocation component from within the flow chart of the business rule to which it belongs; you cannot open it from the System View.

To open a point of view or allocation component:

1. In **System View**, right-click the rule that contains the component, and then select **Open**.

   The business rule is displayed in the Rule Designer.
2. When the rule opens, click the point of view component or double-click the allocation component in the rule's flow chart to open the component.

Deleting a Point of View or Allocation Component

You delete a point of view or allocation component by removing it from the business rule to which it belongs.

Since point of view or allocation components can be used in only one business rule, you delete these components by removing them from the business rule to which they belong.

To delete a point of view or allocation component:

1. In **System View**, right-click the rule that contains the point of view or allocation component, and then select **Open**.
2. In the Rule Designer, select the point of view or allocation component you want to delete in the flow chart.
3. Right-click the point of view or allocation component, and then select **Remove**.

   If the allocation component is within the point of view component, removing the point of view component removes the allocation component.
4. Click  .

Copying and Pasting a Point of View or Allocation Component

You can copy a point of view or allocation component from a business rule, and then you can paste the component into the same or a different business rule.

To copy and paste a point of view or allocation component in a business rule flow chart:

1. In **System View**, right-click the rule that contains the point of view or allocation component, and then select **Open**.

   The business rule is displayed in the Rule Designer.
2. In the **Rule Designer**, right-click the point of view or allocation component you want to copy in the business rule's flow chart, and select **Copy** to copy the component only or **Copy Group** to copy the component and any components within it.
3. Do one of these tasks:
   - To paste the component into the *same* business rule, right-click the location in the flow chart where you want to paste the component, and select **Paste**.
   - To paste the component into a *different* business rule, open the business rule, right-click the location in the flow chart where you want to paste the component, and select **Paste**.
4. Click  .

Saving a Point of View or Allocation Component

You save point of view or allocation component when you save the business rule to which it belongs in the Rule Designer.

Unlike formula components, point of view and allocation components cannot exist independently of the business rule for which they are created.

To save a point of view or allocation component, after you are finished designing it, click  .

Working with Aggregate Storage Formula Components

Create, open, edit, delete, copy, paste, and show usages for aggregate storage formula components.

Related Topics

- Creating an Aggregate Storage Formula Component
- Opening an Aggregate Storage Formula Component
- Editing an Aggregate Storage Formula Component
- Deleting an Aggregate Storage Formula Component
- Copying and Pasting an Aggregate Storage Formula Component
- Copying an Aggregate Storage Formula Component to Another Application or Database
- Showing an Aggregate Storage Formula Component's Usages

Creating an Aggregate Storage Formula Component

An aggregate storage formula component is comprised of formula calculation statements. To create the calculation statements of a formula, you enter or select members and variables. As you create the formula, each of its calculation statements is listed in a row within a grid in the Component Designer.

You can create a formula component from the System View or from within the Rule Designer. A formula component exists as an independent object in the database, so it can be shared amongst business rules.

To create a formula component for an aggregate storage application:

1. In **System View**, right-click **Formulas**, and then select **New**.
2. Enter a formula name and application type.
3. Select an **Application**.

   The application name must be a valid Planning application.
4. Select the **Database**.

> **Note**
>
> If you right-click Formulas, and select New to create a new formula, the New Formula dialog is populated with the application type, the application, and the database you are working in within the System View.

5. Click **OK**.
6. In the Component Designer, in **Properties**, complete these steps:
   a. **Optional:** By default a formula is shared when you create it; you cannot select or clear the Shared check box.

> **Tip**
>
> To create a formula that is *not* shared, open a business rule, then drag a new formula component into the business rule's flow chart. The shared check box is not selected. If you decide to make the formula shared, select the Shared check box.

b. **Optional:** Edit the formula's name by entering a new one of up to 50 characters. (The name defaults from the New Formula dialog.)

c. **Optional:** Enter a description of up to 255 characters for the formula.

d. **Optional:** Enter a caption for the formula. The caption is displayed below the formula in the **Rule Designer** flow chart.

> **Note**
>
> If you do not enter a caption, the component's name is displayed in the flow chart.

e. **Optional:** Enter comments for the formula. For example, you may want to tell users how the formula component should be used.

7. On **Formula**, enter a caption for the formula.
8. **Optional:** From **Offset Member**, if you want the *total* amount of all of the formulas in the formula component to be written to an offset member, or a cross dimension member, click the **Ellipsis** icon to select a member.

> **Note**
>
> You can define an offset member manually within the formulas you create below. The offset defined in the formula component is calculated as the sum of all calculated amounts.

9. To execute aggregate storage custom calculations through MDX Insert, include `USE_MDX_INSERT;` in the comment for the formula.

   `USE_MDX_INSERT` enables the execution of custom calculations and allocations in Aggregate Storage (ASO) cubes. This is applicable only to ASO databases and is a powerful way to perform complex calculations that were traditionally limited to Block Storage (BSO) cubes. See USE_MDX_IMPORT in *Calculation and Query Rerference for Oracle Essbase*.

10. To create a formula statement, click in the first formula statement row to the *left* of the equal sign. Then enter a member or cross dimension member selection, or click **Actions** to select a:
    - Variable (See Working with Variables.)
    - Member (See Adding Members and Functions to a Component.)

> **Note**
>
> To type a cross dimension selection of members, enter each member name, separated by a right arrow. For example, mem1->mem2->mem3.

11. To complete the formula statement, click in the row to the *right* of the equal sign. Then enter a member or cross dimension member selection, or click **Actions** to select a:
    - Variable (See Working with Variables.)
    - Member (See Adding Members and Functions to a Component.)
12. **Optional:** For each formula statement row, click the **Comments** icon to enter comments about the formula statement.
13. In **Usages**, you can view the rules that use the formula component.

> **Note**
>
> None of the information on this tab can be edited.

14. Click  .

Opening an Aggregate Storage Formula Component

You can open an aggregate storage formula component from the System View or from within the Rule Designer flow chart of a business rule that uses the formula component.

To open an aggregate storage formula component, in **System View**, expand **Formulas**, and then do one of these tasks:

- Right-click the formula you want to open, and select **Open**.
- Double-click the formula you want to open.

  The formula component opens in the Component Designer.

> **Note**
>
> To open a formula component within a business rule, open it from within the rule's flow chart by right-clicking the formula component and selecting Open, or by double-clicking the component.

Editing an Aggregate Storage Formula Component

You can edit the formula statements that comprise an aggregate storage formula component and the formula component's comments, caption, name, and description.

To edit an aggregate storage formula component:

1. In **System View**, right-click the formula component, and then select **Open**.
2. In the Component Designer, you can edit any of these properties of a formula component. See Creating an Aggregate Storage Formula Component.

- The caption
- The formula statements
- The name
- The description
- The comments

3. Click  .

Deleting an Aggregate Storage Formula Component

You can delete an aggregate storage formula component only if it is not being used in any business rules. To see if any business rules are using the formula component, you can show the formula component's usages. See Showing an Aggregate Storage Formula Component's Usages.

If the formula component is being used by a business rule, and you no longer need to use the formula component in that rule, remove it from the rule, then delete the formula component. If the formula component is being used in a business rule, and you no longer need the business rule, you can delete the business rule.

If no business rules use the formula component, you can delete the component.

To delete an aggregate storage formula component:

1. In **System View**, right-click the formula, and then select **Show Usages**.
2. Ensure no business rules are using the formula component.

   See Showing an Aggregate Storage Formula Component's Usages.
3. Remove the formula component from any business rules that are using it.

   See Removing a Component from a Flow Chart.
4. Right-click the formula again, and then select **Delete**.
5. Confirm deletion of the formula.

Copying and Pasting an Aggregate Storage Formula Component

You can copy an aggregate storage formula component from a rule and paste it into the same, or a different, rule. You can also copy the contents of the grid within a formula component and paste the contents into the same, or a different, formula component. You cannot copy a formula component and paste it into another formula component or another component type.

To copy and paste an aggregate storage formula component:

1. In **System View**, right-click the rule that contains the formula component, and then select **Open**.
2. In the Rule Designer flow chart, right-click the formula component you want to copy, and select **Copy**.

> **Note**
>
> If the component you want to copy is shared, you can use Edit, Copy Reference to copy the reference to the shared component instead of copying the component itself. (See Copying and Pasting the Reference to a Business Rule Formula or Script Component.)

3. Do one of these tasks:
   - To paste the formula component into the *same* business rule flow chart, right-click in the location of the flow chart, and select **Paste**.
   - To paste the formula component into a *different* business rule flow chart, open the business rule into which you want to paste the component, right-click in the location, and select **Paste**.
4. Click  .

Copying an Aggregate Storage Formula Component to Another Application or Database

You can copy an aggregate storage formula component from one application to another application and database or from one database to another database in the same application.

To copy an aggregate storage formula component:

1. In **System View**, right-click the formula component to copy, and then select **Copy To**.
2. In **Save As**, perform one of these tasks:
   - To copy the formula component to another application, enter the name in **Application**.
   - To copy the formula component to another application and database, enter the names in **Application** and **Database**.
   - To copy the formula component to another database within the same application, enter the name in **Database**.
3. Click **OK**.

   The formula component is displayed in the Formulas node of the application and database to which you copied it.

> **Note**
>
> You may need to refresh the application or database node to which you copy the formula component. Right-click the application or database node, and select Refresh.

Showing an Aggregate Storage Formula Component's Usages

You can see which business rules are using an aggregate storage formula component, and other information about the business rules, by displaying the formula component's usages from the System View.

To show an aggregate storage formula component's usages:

1. In **System View**, right-click the formula whose usages you want to see, and then select **Show Usages**.
2. You can view this information about the formula component:
   - The names of the business rules that are using the formula component
   - The application names of the business rules that are using the formula component
   - The database names of the business rules that are using the formula component
   - The owner of the formula component
   - Whether the business rules that are using the formula component are deployed
   - Whether the business rules that are using the formula component are validated
   - A description of the business rules that are using the formula component

> **Note**
>
> You can also view a formula component's usages from within the Component Designer on the Usages tab.

9

Using Member Selection, Variables, Functions, Smart Lists, and Planning Formula Expressions to Design Components

Related Topics

- About Member Selection, Variables, Functions, and Smart Lists
  Like you use components to design business rules, you use members, variables, and functions to design components.
- Adding Members and Functions to a Component
  Add members to formula components, script components, condition components, and member and data range components.
- Searching for Members
- Working with Variables
  Use variables in components as you design business rules and templates.
- Working with Functions
- Working with Custom Functions
  Use custom functions to perform tasks such as copying and exporting data, removing and adding single or double quotes to a text string, comparing two text strings, and converting dates to other formats.
- Working with Smart Lists
- Working with Planning Formula Expressions
  Use Planning formula expressions in Calculation Manager graphical or script rules.
- Working with Hybrid Aggregation in Essbase
  Hybrid aggregation for block storage databases means that wherever possible, block storage data calculation executes with efficiency similar to that of aggregate storage databases.

About Member Selection, Variables, Functions, and Smart Lists

Like you use components to design business rules, you use members, variables, and functions to design components.

You use member selection in formula, script, condition, member and data range, and member block components to select members and functions that return a list of members. See Adding Members and Functions to a Component.

You use variables to build formulas for formula, loop, and condition components. There are two types of variables: execution variables, which are calculated when the business rule is launched, and replacement variables, which are substituted for more complex formulas or functions. (See Working with Variables.)

You can create these types of variables:

- Global variables for use in all applications belonging to an application type

- Application variables, for use in one application only

  Plan type or database variables, for use in one plan type or database

- Rule variables, for use in one business rule only

Functions are predefined formulas that you can use in loop, condition, and formula components. (See Working with Functions for descriptions of the function types available for Planning applications.) You can use functions to perform calculations like these:

- Converting date strings to numbers
- Calculating the averages value of a member across a range
- Calculating the depreciation of an asset for a time period
- Calculating the period-to-date values of members in the Time dimension

You use a Smart List to select predefined options instead of typing an option in data form cells in Planning.

Adding Members and Functions to a Component

Add members to formula components, script components, condition components, and member and data range components.

Related Topics

- About Adding Members and Functions to a Component
- Adding Members or Functions from One or More Dimensions to a Component
- Removing Members and Functions from a Component

About Adding Members and Functions to a Component

You can add members to formula components, script components, condition components, and member and data range components. You can also add functions that return lists of members to formula, script, and condition components. You can select members and functions from the dimensions in the application to which the component belongs.

Depending on which component you are working with, you can select one or more members or functions from one dimension, or you can select one or more members or functions from multiple dimensions.

- These are the contexts in which you can select a single member for multiple dimensions:
  - In the formula grid of a formula component
  - In a function, where the required parameter is a single member
- You can select multiple members for multiple dimensions when you are defining a variable value whose type is members.
- These are the contexts in which you can select multiple members for a single dimension:
  - In the global range of a business rule
  - In a template where the design-time prompt type is a data intersection
  - In defining a variable value whose type is members
  - In defining a variable limit whose type is member
  - In a loop component

  - In any function where the parameter is members
- These are the contexts in which you can select a single member for a single dimension:
  - In defining a variable value whose type is member
  - In any function where the parameter is member

Adding Members or Functions from One or More Dimensions to a Component

Use the **Member Selector** to add members or functions from one or more diemnsions to a component.

To access the **Member Selector** and add members or functions:

1. In **System View**, right-click a template or business rule, and then select **Open**.
2. When the business rule or template opens, in its flow chart, select a component.

> **Note**
>
> You cannot add a member to a fixed loop component.

3. Click  (**Member Selector** icon).

The **Member Selector** enables you to select members within a dimension. Expand and collapse members within a dimension using the [+] and [-].

The **Member Selector** has these tabs:

- Members
- Functions (if any functions are defined for the dimension)
- Search

All members and functions in the selected dimension are on the **Members** and **Functions** tabs. Use the **Search** tab to search for members or member descriptions.

The members and functions you select are listed under **Selections** on the right. When you are in a component that enables you to select multiple members, you can use **Shift + Click** and **Ctrl + Click** to select contiguous or non-contiguous members.

Members

On the **Members** tab, select a member or members, and click  (Select) to move it to the **Selections** list. You can also use the options in this table to further define your selections.

Table 9-1 Member Selector Buttons

Button	Description
 (Add Special)	**(Planning and Oracle Essbase block storage applications users only)** Select one of these to add additional members or functions related to the member or function you selected on the tab: • Member • Children • iChildren • Descendents • iDescendents • Siblings • iSiblings • Parent • iParent • Ancestors • iAncestors • Relative • Level 0 (Base) • Inclusive **Note:** Planning does not have Level 0 (Base) or Inclusive in its Add Special selections.
 (Select)	Select to move the member or function to the Selections list.
 (Deselect)	Select to remove the member or function from the Selections list.
 (Deselect All)	Select to remove all members and functions from the Selections list.

Functions

On the **Functions**, tab select a function or functions, and enter the required values for the function according to this table:

Table 9-2 Functions and Values

Function	Values to Enter	Description
@ALLANCESTORS	Member Name	Enter the member name or click Member to select a member.
@ANCEST	• Dimension Name • Generation Level Number • Member Name	1. Enter the dimension name you selected in Dimensions. 2. Enter an integer value that defines the generation or level number from which the ancestor value is returned. A positive integer defines a generation number. A value of 0 or a negative integer defines a level number. 3. Enter any member name or member combination.

Table 9-2 (Cont.) Functions and Values

Function	Values to Enter	Description
@ANCESTORS	• Member Name • Generation Level Number • Generation Level Name	1. Enter a member name or member combination. 2. Enter an integer value that defines the absolute generation or level number up to which to include members. A positive integer defines a generation number. A value of 0 or a negative integer defines a level number. 3. Enter a level name or generation name up to which to include members in the selection.
@ATTRIBUTE	Attribute Member Name	For the dimension you entered, enter the attribute member name or member combination you want to include in the selection.
@CHILDREN	Member Name	Enter a member name or member combination, or a function that returns a member or member combination.
@CURRMBR	Dimension Name	Enter the dimension name.
@DESCENDANTS	• Member Name • Generation Level Number • Generation Level Name	1. Enter a member name or member combination. 2. Enter an integer value that defines the absolute generation or level number up to which to include members. A positive integer defines a generation number. A value of 0 or a negative integer defines a level number. 3. Enter a level name or generation name up to which to include members in the selection.
@GENMBRS	• Dimension Name • genName	1. Enter a dimension name. 2. Enter a generation name from dimName. A positive integer defines a generation number.
@IALLANCESTORS	Member Name	Enter a member name or member combination, or a function that returns a member or member combination.
@IANCESTORS	• Member Name • Generation Level Number	1. Enter a member name or member combination, or a function that returns a member or member combination. 2. Enter an integer value that defines the absolute generation or level number up to which to include members in the selection. A positive integer defines a generation number. A value of 0 or a negative integer defines a level number.
@ICHILDREN	Member Name	Enter a member name or member combination, or a function that returns a member or member combination.
@IDESCENDANTS	• Member Name • Generation Level Number	1. Enter a member name or member combination, or a function that returns a member or member combination. 2. Enter an integer value that defines the absolute generation or level number up to which to include members in the selection. A positive integer defines a generation number. A value of 0 or a negative integer defines a level number.
@ILSIBLINGS	Member Name	Enter a member name or member combination, or a function that returns a member or member combination.

Table 9-2 (Cont.) Functions and Values

Function	Values to Enter	Description
@IRDESCENDANTS	• Member Name • Generation Level Number	1. Enter a member name or member combination, or a function that returns a member or member combination. 2. Enter an integer value that defines the absolute generation or level number up to which to include members in the selection. A positive integer defines a generation number. A value of 0 or a negative integer defines a level number.
@IRSIBLINGS	Member Name	Enter a member name or member combination, or a function that returns a member or member combination.
@ISIBLINGS	Member Name	Enter a member name or member combination, or a function that returns a member or member combination.
@LEVMBRS	• Dimension Name • Level Name	1. Enter a dimension name. 2. Enter a level name or an integer value that defines the number of a level. The integer value must be 0 or a positive integer.
@LIST	Argument	Enter a list of arguments that will be collected and treated as one argument so they can be processed by the parent function. Arguments can be member names, member combinations, member set functions, range functions, and numeric expressions.
@LSIBLINGS	Member Name	Enter a member name or member combination, or a function that returns a member or member combination.
@MATCH	• Member, Generation • genName • Pattern to Match	1. Enter the default or user-defined name of the member on which to base the selection. The system searches the member names and alias names of the specified member and its descendants. 2. Enter the default or user-defined name of the generation on which to base the selection. The system searches all member names and member alias names in the generation. 3. Enter the character pattern to search for, including a wildcard character (* or ?). ? substitutes one occurrence of any character. You can use ? anywhere in the pattern. * substitutes any number of characters. You can use * only at the end of the pattern. To include spaces in the character pattern, enclose the pattern in double quotation marks ("").
@MEMBER	String	Enter a string (enclosed in double quotation marks) or a function that returns a string.
@MERGE	• List 1 • List 2	1. Enter the first list of members to be merged. 2. Enter the second list of members to be merged.
@PARENT	• Dimension Name • Member Name	1. Enter the dimension name. 2. Enter a member name or member combination, or a function that returns a member or member combination, to be combined with the parent returned.

Table 9-2 (Cont.) Functions and Values

Function	Values to Enter	Description
@RANGE	• Member Name • Range List	1. Enter a member name or member combination, or a function that returns a member or member combination, to be combined with the parent returned. 2. Enter a member name, a comma-delimited list of member names, member set functions, or range functions. If rangeList is not specified, the system uses the level 0 members from the dimension tagged as Time.
@RDESCENDANTS	• Member Name • Generation Level Number	1. Enter a member name or member combination, or a function that returns a member or member combination, to be combined with the parent returned. 2. Enter an integer value that defines the absolute generation or level number down to which to select the members. A positive integer defines a generation number. A value of 0 or a negative integer defines a level number.
@RELATIVE	• Member Name • Generation Level Number	1. Enter a member name or member combination, or a function that returns a member or member combination, to be combined with the parent returned. 2. Enter an integer value that defines the absolute generation or level number down to which to select the members. A positive integer defines a generation number. A value of 0 or a negative integer defines a level number.
@REMOVE	• List 1 • List 2	1. Enter the first list of members to be merged. 2. Enter the second list of members to be merged.
@RSIBLINGS	Member Name	Enter a member name or member combination, or a function that returns a member or member combination.
@SHARE	Range List	Enter a comma-delimited list of members, functions that return members, or ranges of members. All the members in rangeList must be from the same dimension.
@SIBLINGS	Member Name	Enter a member name or member combination, or a function that returns a member or member combination.
@UDA	• Dimension Name • User Defined Attribute	1. Enter the name of the dimension with which the user-defined attribute is associated. 2. Enter the name of the user-defined attribute as it appears in the database outline.
@WITHATTR	• Dimension Name • Operator • Value	1. Enter the name of the attribute dimension. 2. Enter the operator specification, enclosed in quotation marks (""). 3. Enter a value that, in combination with the operator, defines the condition that must be met. The value can be an attribute member specification, a constant, or a date-format function (that is, @TODATE).

Table 9-2 (Cont.) Functions and Values

Function	Values to Enter	Description
@XRANGE	• Member Name 1 • Member Name 2	1. Enter a member name, member combination, or function that returns a single member. 2. Enter a member name, member combination, or function that returns a single member. If mbrName1 is a cross-dimensional member (such as Actual->Jan), then mbrName2 must be also, and the dimension order must match the order used in mbrName1.

Search

On the **Search** tab, do these tasks to search for a member or members:

1. From **Dimensions**, select a dimension in which you want to search for a member.
2. Under **Find**, select a type of member to search for, a member name or its description.
3. Enter the name of the member, or its description, to search for, or to display all members in the dimension, accept the default wildcard ().
4. Select **Search** to search for a member you enter in the field. (See Searching for Members in the Member Selector.)
5. Select **Advanced Search** to access advanced search options. (See Searching for Members in the Member Selector by Name, Alias, or Property.)
6. Select the member or members, and click the **right arrow** to move them to the **Selections** list.

Removing Members and Functions from a Component

You can remove *members* from formula, script, condition, and member and data range components. You can remove *functions* from formula, script, and condition components.

When you remove members and functions from a component, they are not deleted from the database. To remove members and functions from a shared component, you must make the component not shared first.

To remove members or functions from a component:

1. In **System View**, right-click a template or business rule, and then select **Open**.
2. When the business rule or template opens, in its flow chart, select the component that contains the member or function to remove.
3. Click  (**Member Selector** icon).
4. In the **Member Selector**, in **Selections**, select a dimension to remove all members from that dimension, or select a member to remove only that member from the dimension.
5. Use the left arrow to move the member or function from **Selections** to **Members** or **Functions**.
6. Click **OK**, and then  (Save).

Searching for Members

Related Topics

- Searching for Members in the Member Selector
- Searching for Members in the Member Selector by Name, Alias, or Property

Searching for Members in the Member Selector

You can search for members within the Member Selector.

To access the Member Selector and search for members:

1. In **System View**, right-click a template or business rule, and then select **Open**.
2. When the template or business rule opens, in its flow chart, select a component.
3. Click  (**Member Selector** button).
4. In the **Member Selector**, the dimension name, its alias (if there is one), and the count (number of members in the dimension) is displayed on the **Members** tab. By default, the outline is collapsed.
5. From the **Search** tab, select whether to search by **Name** or **Alias**, enter the name or alias to search for, and click the **Find** button.
6. **Optional:** Click  (**Advanced Search**) to search for the member by its name, alias, or one of its properties. See Searching for Members in the Member Selector by Name, Alias, or Property.
7. Select the member or members, and click  (**Right Arrow**) to move them to the **Selections** list, and the click **OK**.

Searching for Members in the Member Selector by Name, Alias, or Property

You can use Advanced Search within the Member Selector to search for a member by its name, its alias, or one of its properties.

To search for a member by name, alias, or property:

1. From the **Member Selector**, select the **Search** tab, and click **Advanced Search**.
2. In **Find Members**, from **Search By**, select one of these options:
   - **Name**, to search for the member by its name. Then go to step 4.
   - **Alias**, to search for the member by its alias. Then go to step 4.
   - **Property**, to search for the member by one of its properties. Then go to step 3.
3. If you selected **Property**, enter or select a **Property Name**.
4. Enter a value for the name, alias, or property.
5. Click **OK**.

   If the alias, name or property is found, it is shown in the **Results**. The outline is not shown, only the members that are found are shown.

> **Note**
>
> When you search for members by alias, all members that have aliases matching the search criteria, including members with aliases in other languages, are shown in Results. The outline is not shown, only the members that are found are shown. However, only aliases for members in the language being used are displayed in the Member Selector.

6. **Optional:** If more than one member matches your search criteria, use the up and down arrows or the scroll bar if it is available to move up and down to locate all members that match your search criteria.

Working with Variables

Use variables in components as you design business rules and templates.

Related Topics

- About Variables
- Creating a Variable
- Entering Runtime Prompt Variables
- Selecting a Variable
- Editing a Variable
- Deleting a Variable
- Refreshing Variables
- Copying a Variable
- Finding and Replacing Text in the Variable Designer
- Showing the Usages of a Variable

About Variables

Variables assume values that you define for them. You use them in components as you design business rules and templates.

You can create variables in the following ways:

- Click  to launch the Variable Designer.
- Create variables from within a rule, script, formula, or template, anywhere the Variable Selector dialog box is available.
  For example:
  - Open a rule, and then drag in a Member Range component.
  - Click  next to a dimension, and then select **Variable**.
  - In the **Select Variable** dialog box, click **Create** to create a variable.

When you create a variable, the variable is created at the level you choose: global, application, plan type, or business rule. If the same named variable is created at each level, the lowest level variable is used in the rule. For example, if you create a Global variable named:vMonth and a plan type variable named Month, the rules will use the variable from the plan type.

There are two types of variables:

- **Execution**—When the business rule is launched, the calculation defined for the variable is performed. You can use execution variables in script components or fixed loop components.
- **Replacement**—When you are designing or launching the business rule, the variable is substituted within a calculation. You can use replacement variables in any component.

  You can create several types of execution and replacement variables. The variables you can create differ depending on the application type and whether you are creating an execution or a replacement variable.

> **Note**
>
> For non-groovy rules, variables of type *member* or *members* are the only supported types of variables for Planning cubes of type *Aggregate Storage Option* (ASO).

You can create variables that prompt users to enter information when they launch a business rule. These runtime prompt variables prompt users for such information as members, text, dates, or numbers. The prompts tell users what type of data is expected.

For example:

- Select a month.
- Enter the expected number of customer visits per quarter.
- Specify what percentage change in earnings you expect next month.

There are up to four database objects with which you can associate a variable, depending on the application type for which you are creating a variable. A variable can exist in multiple objects simultaneously and can have the same name in each object.

Creating a Variable

To create a variable:

1. In **System View**, **Custom View**, **Filter View**, or **Deployment View**, click  .
2. In the **Variable Navigator**, expand the application type.

   For example, expand **Planning**.
3. Choose the level at which to create the variable.
   - **Global**—Right-click **<Global>**, and then select **New** to create a variable that can be used in any application of the same application type.
   - **Application**—Right-click an application, and then select **New** to create a variable that can be used in that application only.
   - **Plan or Database**—Right-click a plan type or database, and then select **New** to create a variable that can be used in that plan type or database only.
   - **Business Rule**—Right-click a business rule, and then select **New** to create a variable that can be used in that rule only.
4. Select the type of variable to create:

- **Replacement**—When you are designing or launching the business rule, the variable is substituted with a calculation. You can use replacement variables in any component.

  To create a Replacement variable, on the **Replacement** tab, select **Actions**, then **New**, and then enter the following information:

  - **Name**—Variable name
  - **Description**—Description of the variable
  - **Group**—To include this variable in a group, enter the group name. The group name is displayed under the **Group** column after you save and refresh the variable.
  - **Type**—Click the drop-down and select a type.
  - **RTP**—If this is a runtime prompt variable:
    - Select **RTP** and enter the text in to display each time the variable is used.
    - Enter a **Default Value** if desired.
    - Select **Use Last Entered Value** if to display the last value entered for the prompt as the default value the next time the prompt occurs.
- **Execution**—When the business rule is launched, the calculation defined for the variable is performed. You can use execution variables in script components or fixed loop

  To create an Execution variable, on the **Execution** tab, select **Actions**, then **New**, and then enter the following information:

  - **Name**—Variable name
  - **Group**—To include this variable in a group, enter the group name. The group name is displayed under the **Group** column after you save and refresh the variable.
  - **Value**—Variable value
    - For numeric variables, see Entering Variable Values for a Numeric Variable
    - For string variables, see Entering Variable Values for a String Variable
    - For member range variables, see Entering Variable Values for a Member or Members Variable

5. Click 

Entering Variable Values for a Numeric Variable

A numeric variable can be a Planning replacement variable.

To enter values for a numeric variable:

1. From **Type**, select **Numeric**.
2. To use a Smart List, create the variable at the application, plan type, or rule level. (You cannot use the Smart List at the global level.) Then click in the Smart List box, click on the dropdown list, and select a Smart List.

   See *Administering Planning for Oracle Planning and Budgeting Cloud Service* for this release. For example, you can set up an integer Smart List for a reporting cycle that has values 1-5, for Yearly (1), Quarterly (2), Monthly (3), Daily (4), and Hourly (5). A user can select "Monthly" and the number three is stored in the database. This prevents users from having to remember the numbers.

You can also set up a string of text or a date as the value for the Smart List.

> **Note**
>
> You must select a Planning application that supports the use of Smart Lists.

3. To use a numeric runtime prompt, leave the **Smart List** box empty, then go to the next step.
4. **Optional:** In **Limits**, select a limit for the variable.
5. **Optional:** Enter a default value for the variable.
6. By default, **RTP** is selected. If you do not want to create a runtime prompt for this variable, clear **RTP**.

> **Note**
>
> If you do not select RTP, you must enter a default value for the variable.

7. If you selected **RTP**, enter the runtime prompt text you want to display as the default value for users.
8. Specify whether missing data values are allowed.
9. Click  .

To enter values for a Oracle Hyperion Financial Management replacement or execution numeric variable:

Entering Variable Values for a String Variable

A string variable can be a Planning replacement variable. A string variable must be alphanumeric and be no more than 255 characters. It can contain a null value, but cannot contain a leading & (ampersand) character in the value.

To enter values for a replacement string variable:

1. From **Type**, select **String**.
2. In the **Value** table, enter a value for the variable.
3. By default, **RTP** is selected. If you do not want to create a runtime prompt for this variable, clear **RTP**.

> **Note**
>
> If you do not select RTP, you must enter a default value for the variable.

4. f you selected **RTP**, enter the runtime prompt text you want to display for users.
5. Click  .

To enter values for a Oracle Hyperion Financial Management replacement or execution string variable:

Entering Values for an Array Variable

An array variable can be a Planning execution variable. Arrays contain a list of values that can be multidimensional.

Typically, arrays are used to store variables as part of a member formula. The size of the array variable is determined by the number of members in the corresponding dimension. For example, if the Scenario dimension has four members, the following command creates an array called Discount with four entries. You can use more than one array at a time.

```
ARRAY Discount[Scenario];
```

To enter values for an array variable:

1. Select the **Array** check box to make this execution variable an Array. After you select **Array**, you must select a dimension from the dropdown list, or it defaults to a dimension in the plan type.
2. In the **Value** field, enter a value creating the variable at the plan level.
3. **Optional**: Enter a group for the variable.
4. Click  .

Entering Values for a Member Range Variable

A member range variable can be a Planning replacement variable. The member range variable should contain a range of members.

To enter values for a member range variable:

1. From **Type**, select **Member Range**.
2. In the **Variable grid**, perform these steps:
   a. For each dimension in the table for which you want to select limits for the member range, click in the limit field and enter a limit. (The dimensions that display are the dimensions that belong to the application for which you are creating the variable.)

      At the global level, if you select the Dimension Type option, only the standard dimension types are shown. If you select the Dimension Names option, you can type in any dimension name.

> **Note**
>
> You must select RTP before you enter text in the Limits field.

> **Note**
>
> When using a function for the Limit, Planning functions are recommended instead of Oracle Essbase functions. In some cases, Essbase functions do not return the expected members; for example, when the evaluation of the Essbase function includes dynamic members. For the Limit in a member range variable, use "ILvl0Descendants("Mbr Name") " instead of the Essbase function "@Relative("Mbr Name", 0)".

b. Enter or use the Member Selector to select default values for the member range, or if the variable is a runtime prompt, you can leave the default value empty. You can select multiple members and functions for each dimension listed.

c. Select **RTP** for each dimension listed if you want the variable to prompt users for information when it is launched.

> **Note**
>
> If you do not select RTP, you must enter a value for the variable.

d. For each dimension for which you selected **RTP**, enter the runtime prompt text.

e. In the RTP text box above the grid, enter the runtime prompt text you want to display for users each time the variable is launched for that dimension.

3. Click  .

Entering Variable Values for a Cross Dimension Variable

A cross dimension variable is a Planning replacement variable. It contains a member from multiple dimensions that enables you to launch business rules across dimensions.

To enter values for a cross dimension variable:

1. From **Type**, select **Cross Dimension**.
2. **Optional:** For each dimension in the table for which you want to select limits for the variable, click in the limit field and enter a limit. (The dimensions that display are the dimensions that belong to the application for which you are creating the variable.)

> **Note**
>
> - You must select RTP before you enter text in the Limits field.
> - When using a function for the Limit, Planning functions are recommended instead of Oracle Essbase functions. In some cases, Essbase functions do not return the expected members; for example, when the evaluation of the Essbase function includes dynamic members. For the Limit in a cross dimension variable, use "ILvl0Descendants("Mbr Name") " instead of the Essbase function "@Relative("Mbr Name", 0)".
> - You can use a function, but the function must return a single member from the dimension to be a valid selection.

3. Enter or use the Member Selector to select a value for the variable. You can select a member or a function.
4. Enter the runtime prompt text.
5. Click  .

Entering Variable Values for a Dimension Variable

A dimension variable is a Planning replacement variable. This variables contains a dimension that you select.

To enter values for a dimension variable:

1. From **Type**, select **Dimension**.
2. In the **Variable Grid**, select a dimension. The dimensions that display are the dimensions that belong to the application for which you are creating the variable.
3. If you selected **RTP**, enter the runtime prompt text you want to display for users.
4. Click  .

Entering Variable Values for a Member or Members Variable

The member and members variables are Planning replacement variables. These variables contain a member or multiple members from a dimension that you select.

To enter values for a member or members variable:

1. From **Type**, select **Member** or **Members**.
2. In the **Variable Grid**, select a dimension. The dimension that displays is the dimension that belongs to the application for which you are creating the variable.
3. Enter or use the Member Selector to select limits for the variable. You can select members only from the dimension you select in step 2. You can also select functions. See Working with Functions.

> **Note**
>
> When using a function for the Limit, Planning functions are recommended instead of Oracle Essbase functions. In some cases, Essbase functions do not return the expected members; for example, when the evaluation of the Essbase function includes dynamic members. For the Limit in Member or Members variable, use "ILvl0Descendants("Mbr Name") " instead of the Essbase function "@Relative("Mbr Name", 0)".

4. Enter or use the Member Selector to select a default value for the variable. You can select one member or function for a *member* variable and multiple members and functions for a *members* variable.
5. By default, **RTP** is selected. If you do not want to create a runtime prompt variable, clear **RTP**.

> **Note**
>
> If you do not select RTP, you must enter a value for the variable.

6. If you selected **RTP**, enter the runtime prompt text you want to display for users.
7. Select **File**, and then **Save**.

Entering Variable Values for Percent Variables

The Percent variable is a Planning replacement variable. The Percent variable is also a Planning execution variable. This variable contains a percentage that you specify.

To enter values for a percent variable:

1. From **Type**, select **Percent**.
2. In the **Variable Grid**, click in **Limits** to define minimum and maximum values for the variable.
3. Enter a numeric value for the variable.
4. y default, **RTP** is selected. If you do not want to create a runtime prompt, clear **RTP**.

> **Note**
>
> If you do not select RTP, you must enter a value for the variable.

5. If you selected **RTP**, enter the runtime prompt text you want to display for users.
6. Select whether to allow missing data values.
7. Select **File, Save**.

Entering Values for Integer Variables

The Integer variable is a Planning replacement variable.

To enter values for an integer variable:

1. From **Type**, select **Integer**.
2. To use a Smart List, create the variable at the application, plan type, or rule level. (You cannot use the Smart List at the global level.) Then click in the Smart List box, click on the dropdown list, and select a Smart List.

   For example, you can set up an integer Smart List for a reporting cycle that has values 1-5, for Yearly (1), Quarterly (2), Monthly (3), Daily (4), and Hourly (5). A user can select "Monthly" and the number three is stored in the database. This prevents users from having to remember the numbers.

   You can also set up a string of text or a date as the value for the Smart List.

> **Note**
>
> You must select a Planning application that supports the use of Smart Lists.

3. To use an integer runtime prompt, leave the Smart List box empty, then go to the next step.
4. **Optional:** In the **Value** table, click in **Limits** to define minimum and maximum integers values for the variable.
5. **Optional:** Click in the default value and enter an integer for the variable.
6. By default, **RTP** is selected. If you do not want to create a runtime prompt for this variable, clear **RTP**.

If you do not select RTP, you must enter a default value for the variable.

7. If you selected **RTP**, enter the runtime prompt text you want to display for users.
8. Select whether to allow #Missing values.
9. Select **File, Save**.

Entering Variable Values for String as Number Variables

The String as Number variable can be a Planning replacement or execution variable.

To enter values for a string as number variable:

1. From **Type**, select **String as Number**.
2. **Optional:** In the **Value** table, click in **Limits** to define minimum and maximum values for the variable. The minimum and maximum values should be entered as numbers in the format of YYYYMMDD.
3. **Optional:** Enter a numeric value for the variable.
4. By default, **RTP** is selected. **RTP** is required for StringAsNumber variables, so you cannot clear the **RTP** check box.
5. Select whether to allow #Missing values.
6. Select **Use last entered value**, to allow users to use the last value they entered.
7. Click  .

Entering Variable Values for Date as Number Variables

The Date as Number variable can be a Planning replacement or execution variable.

To enter values for a date as number variable:

1. From **Type**, select **Date as Number**.
2. **Optional:** In the **Value** table, click in **Limits** to define minimum and maximum values for the variable. The minimum and maximum values should be entered as numbers in the format of `YYYYMMDD`.
3. **Optional:** Enter a numeric value for the variable. For dateasnumber variables, enter a date in the numeric format of `YYYYMMDD`.
4. By default, **RTP** is selected. If you do not want to create a runtime prompt, clear **RTP**.

   If you do not select RTP, you must enter a value for the variable.
5. If you selected **RTP**, enter the runtime prompt text you want to display for users.
6. Select whether to allow #Missing values.
7. Select **Use last entered value**, to allow users to use the last value they entered.
8. Click  .

Entering Runtime Prompt Variables

Note

- You cannot add a RUNTIMESUBVARS section to the script of an Oracle Essbase business rule in Calculation Manager. Rule level runtime prompt variables that are created in Calculation Manager are converted into RUNTIMESUBVARS only when you deploy Essbase applications.
- If you launch a business rule with runtime prompts in Administration Services, MaxL, or any component that can launch a calc script, the runtime prompts in that business rule must have default values.
- You can enter or edit values for runtime prompt variables when validating, debugging, deploying, analyzing, or launching a rule in Calculation Manager. You can also enter or edit runtime prompt variables when you are validating or deploying business rulesets. If the runtime prompt contains member limits, the validation checks only for valid member names (it does not validate whether the member is within the limit). *Numeric* and *Integer* type variables are validated for runtime prompt limits.
- For information on desiging runtime prompts to honor Approvals security for members, see About Runtime Prompts in Approvals Security in *Administering Planning*.

To enter values for runtime prompt variables:

1. When you validate, debug, deploy, analyze, or launch a business rule, or validate or deploy a business ruleset, if there are no errors, the **Enter RTP Values** dialog box is displayed.

   When you validate, debug, deploy, or analyze a business rule, or validate or deploy a business ruleset, the Enter RTP Values dialog is displayed only if there are missing values for one or more of the runtime prompt variables the business rule (or business ruleset) is using. If all runtime prompt variables have values, then Enter RTP Values dialog is not displayed.

   When you launch a business rule, the Enter RTP Values dialog is displayed each time, regardless of whether the runtime prompt variables have values. If there are values, those values are displayed by default in the Enter RTP Values dialog.
2. For each of the runtime prompts listed, enter a value or select one .
3. **Optional:** if you are working with a business rule, select the **Apply values to the rule** check box so the values you provide are dynamically updated in the variable's value and visible in the **Value** column of the **Variables** tab within the Rule Designer.

   This check box is not available if you are validating a business rule from the System View.
4. Click **OK**.
5. If there are validation errors, fix them, and repeat the task for which you want to enter runtime prompt variable values.

Selecting a Variable

You can select a variable from various locations. You can select variables as you create components from within the Component Designer, as you create design-time prompts from within the Template Designer, and other locations in Calculation Manager.

To select a variable:

1. Do one of these tasks:
   - Right-click the template that contains the component you want to add a variable to, and select **Open**.
   - Right-click the business rule that contains the component you want to add a variable to, and select **Open**.
2. When the business rule or template opens, in its flow chart, select the component for which you want to insert a variable.
3. In the tabs below the flow chart, do one of these tasks:
   - For member range components, click in the dimension field, select the **Actions** icon, and select **Variable** to select a member range variable. Or click on the **Variable Selector** to select a member range variable.
   - For data range and fixed loop components, in the **Variable** field, select the **Variable** icon.
   - For formula components, click the **Actions** icon, and select **Variable**.
   - For script components, click the **Insert a Variable** icon.
   - For condition components, launch the **Condition Builder**, click the **Actions** icon, and select **Variable**.
4. In **Select Variable**, do one of these tasks:
   - To create a new variable, click **Create** to access the **Variable Designer**. See Creating a Variable.
   - To select an existing variable, from **Category**, select the level that contains the variable you want to use. You can select:
     - Global: The variable was created at the global level and can be used by all applications under this application type.
     - Application: The variable was created at the application level, and is only shown for this application.
     - Plan Type or Database: The variable was created at the plan type or database level and is only shown in this plan type or database.
     - Rule: The variable was created at the rule level and is only shown for the rule in which it was created.

> **Note**
>
> For member range, member block, formula, fixed loop, and condition components, the variables shown are restricted to the expected type of variable that the component uses, so all of the available variables for the selected scope are not shown by default.To see all of the available variables in the selected scope, select the Show all variables check box.

5. On **Replacement** or **Execution**, select one or more variables to insert into the component.
6. Click **OK**, and the  .

Editing a Variable

You can edit any property of a variable from the Variable Designer. When you make changes to a variable, if that variable is used in a rule, you should open the rule, save it, validate it, and redeploy it. See Validating and Deploying.

Deleting a Variable

You can delete a variable or variables from within the Variable Designer if they are not used in any components or member formulas. If a variable is used in a component, you must remove the variable from the component before you delete the variable.

To delete a variable:

1. In **System View** or **Filter View**, click the **Variable Designer** icon.
2. In the Variable Navigator, expand the application type and the application.
3. Do one of these tasks:
   - If the variable is a global variable, select **<Global>**.
   - If the variable is an application variable, select the application with which the variable is associated.
   - If the variable is a plan type or database variable, select the plan type or database with which the variable is associated.
   - If the variable is a business rule variable, select the business rule with which the variable is associated.

   Any variables associated with the application type, the application, the calculation type, plan type, or database, and the business rule are displayed on **Replacement** or **Execution**.
4. On **Replacement** or **Execution**, right click the variable you want to delete, and select **Delete**.
5. In **Delete Confirmation**, select **Yes** to confirm deletion of the variable.

Refreshing Variables

You can refresh the list of variables in the Variable Navigator to see the most current list after you add, delete, or make changes to variables.

To refresh the list of variables in the Variable Navigator:

1. In **System View** or **Filter View**, click the **Variable Designer** icon.
2. In the Variable Navigator, create, edit, or delete a variable.
3. Above the **Replacement** or **Execution** tab, click the **Refresh** icon.

Copying a Variable

You can copy a variable to the same or a different variable scope (that is, global level, application level, consolidation, plan, or database level, or rule level) using copy and paste. If the variable you are copying has the same name as a variable in the location into which you are copying it, you can give the variable you are copying a new name, skip copying the variable, or overwrite the contents of the variable.

To copy and paste a variable:

1. In **System View** or **Filter View**, click the **Variable Designer** icon.
2. In the Variable Navigator, expand Planning, and select **Global** or the application, plan type, or business rule associated with the variable you want to copy.
3. Right-click the variable you want to copy, and select **Copy**.
4. Right-click the location or variable into which you want to paste the copied variable, and select **Paste**. (For example, if you are copying a Planning global variable, you may copy it as another Planning global variable or as a plan type variable)
   - If you are trying to copy a variable and paste it into a location that contains a variable with the same name, the **Resolve Conflicts** dialog is displayed. Perform one of these tasks:
     - Give the variable a new name. (You cannot have two variables with the same name in the same location.)
     - Specify to skip copying the variable. (The contents of the copied variable are not pasted to the new location.)
     - Specify to overwrite the variable. (The contents of the copied variable are pasted to the new location and overwrite the contents of the copied to variable.)
   - If you are trying to copy a variable and paste it into a location that does not contain a variable with the same name, the variable is pasted to the new location.

Finding and Replacing Text in the Variable Designer

You can search for and replace text in variables within the Variable Designer. You can search through variables of any scope: (global, application, plan or database, or business rule).

You can search for a variable by entering its name in the Find feature of the Variable Designer. You can also search for a text string in a variable. (For example, you can search for a default value used in the variable.)

By default, Calculation Manager searches using the variable's default value, any limits defined for it, and any prompt text. If you include the basic properties of the variable in the search, Calculation Manager searches using the variable name, the group, and the description.

You can replace all instances of a variable or text string, or you can replace a selected instance. When you replace text, and the Limits field contains a variable of the type smartlist or number, these are excluded from the replace operation. Replacing these fields may leave the variable definition in an incorrect state (for example, the variable may have an invalid smart list name or a default value that is not within the specified limits.)

To find text in the Variable Designer:

1. From any view, click the **Variable Designer** icon.
2. In the **Variable Navigator**, right click the application type, Global (Planning and Oracle Hyperion Financial Management users only), the plan type or database, or the business rule in which you want to search, and select **Find**.
3. From **Any Text**, select:
   - **Starts With**, to display only variables whose names start with characters you specify
   - **Ends With**, to display only variables whose names end with characters you specify
   - **Contains**, to display only variables whose names contain characters you specify
   - **Matches**, to display only variables whose names match characters you specify
4. In the **Search** field, enter the text of the variable for which to search.
5. Select one or more of these options:
   - Select **Ignore case** if you do not need the case of the text you are searching for to match the case of the text you enter in the **Search** field.
   - Select **Include Basic Properties**, to include the variable name, description, and group in the search.

     Ignore case and Include Basic Properties are selected by default when you launch the Variable Designer. If you clear these check boxes, the check boxes remain cleared until you close and reopen the Variable Designer.
   - Select **Include variables in child scopes** to search for the variable in levels under the selected node. If you are searching for variables at the *application type* level (for example, Oracle Essbase or Planning), this check box is selected by default and cannot be changed. With this option selected, variables for the applications, plan types, consolidation types, or databases, and business rules are displayed. In addition to columns for the variable name, description, default value, group, and owner, an Application column, Plan Type column, and a Rule column are also displayed in the list of variables.

     If you are searching for variables at the *application* level, this check box can be selected or cleared. When you select this option, variables for the application, its plan types, consolidation types, or databases, and its business rules are displayed. In addition to columns for the variable name, description, default value, group, and owner, a Plan Type column and a Rule column are also displayed.

     This option is not available when searching for global variables in Planning. When you search on global variables, only global variables display.
6. Click **OK**.

   If the text you search for is found, the variables in which it is found are listed on the Replacement or Execution tabs.

After you find text, you can replace one or more instances of the text.

To replace text in the Variable Designer, do either of these tasks:

- To replace a selected instance of a text string:
  1. Select the variable in which you want to replace the text string.
  2. Select **Actions** and then **Replace Selected**.
  3. In the **Replace Selected** dialog box, in **Replace with**, enter the text with which to replace the text string.

4. Click **Replace Selected**.

- To replace all instances of the text string:
  1. Select **Actions** and then **Replace All**.
  2. In the **Replace All** dialog box, in **Replace with**, enter the text with which to replace the text string.
  3. Click **Replace All**.

> **Note**
>
> Any options you selected while searching for the text string are selected by default in the Find area of the Replace Selected and Replace All dialog boxes and cannot be changed. For example, if you cleared the Ignore case check box when you searched for the text string, the Ignore case check box is cleared in the Replace Selected/ Replace All dialog box and cannot be changed.

Showing the Usages of a Variable

You can display the business rules that use variables. When you show the usages of a variable, this information is displayed:

- The names of the business rules that are using the variable
- The application names of the business rules that are using the variable
- The plan types of the business rules that are using the variable
- The owners of the business rules that are using the variable
- Whether the business rules that are using the variable are deployed
- Whether the business rules that are using the variable are validated
- A description of the business rules that are using the variable

To show the usages of a variable:

1. From the System or Filter view, click the **Variable Designer** icon.
2. In the Variable Navigator, select the database object that contains the variable whose usages you want to see. The variables defined for that object are displayed on the **Replacement** and **Execution** tabs in the Variable Designer.
3. Right-click the variable whose usages you want to see, and select **Show Usages**.
4. After you review the information, click **OK**.

Working with Functions

Related Topics

- About Functions
  Functions define member formulas that return data values or members.
- Inserting Functions into Components
  The function types you can select from differ depending on the application type and component type with which you are working.

- Essbase Functions Supported by Calculation Manager
  The following Oracle Essbase functions are supported by Calculation Manager in Block Storage applications.

About Functions

Functions define member formulas that return data values or members.

For example, you can use functions (and mathematical and logical operators) to return a list of siblings, parents, or children of a member you specify, to return a list of data values that are greater than or less than a value you specify, or to allocate data values from a member you specify. When you select a function, you are prompted to enter the correct parameters.

If you are working with Planning applications, you use functions in formula, script, condition, and member range components.

Following is a list of the types of functions you can use in Planning and Oracle Essbase block storage components. (See the Oracle Essbase Technical Reference for a complete list ,and descriptions of the functions.)

- Boolean
- Relationship
- Calculation Operators
- Control Flow
- Data declarations
- Functional
- Mathematical
- Member Set
- Range (Financial)
- Allocation
- Forecasting
- Statistical
- Date & Time
- Miscellaneous
- Custom

> **Note**
>
> Functions are available in the Member Selector and the Function Selector.
>
> You use only Member Set functions in Essbase aggregate storage applications and components, including those used in Oracle General Ledger. (See the Oracle Essbase Technical Reference for a complete list and descriptions of the Member Set functions.)

Inserting Functions into Components

The function types you can select from differ depending on the application type and component type with which you are working.

You can insert functions into formula, script, condition, and member range components, if you are working with a Planning application.

To insert a function into formula, script, condition, or member range component:

1. Do one of these tasks:
   - Open the business rule that contains the component into which you want to insert a function. Then select the component for which you want to insert a function in the business rule's flow chart.
   - Open the template that contains the component into which you want to insert a function. Then select the component for which you want to insert a function in the template's flow chart.
   - Open the formula component into which you want to insert a function.
   - Open the script component into which you want to insert a function.
2. Do one of these tasks:
   - To insert a function into a formula component, on the **Formula** tab, click in the **Formula** row, select the **Actions** icon, and select **Functions**.
   - To insert a function into a script component, on the **Script** tab, click the **Insert a function and its parameters** icon.
   - To insert a function into a member range component, on the **Member Range** tab, click in the **Value** column for a dimension, select the **Actions** icon, and select **Functions**.
   - To insert a function into a condition component, on the **Condition** tab, click the **Condition Builder** icon. In the Condition Builder, from **Formula**, **Function**, or **Value**, select the **Actions** icon, and select **Functions**.
3. In the **Function Selector**, do one of these tasks:

   > **Note**
   >
   > Depending on the context in which you open the Function Selector, the available function types may be more limited than the function types described in Working with Functions.

   - If you can select function types from **Category**, select one, or select **All Functions** to display functions for all function types. The functions for the category, or all functions, are displayed in a list below the Category selection.
   - If you cannot select among function types in **Category**, proceed to 4.
4. Select a function from the list of functions.
5. Enter parameters for the function.
6. Click **OK**.

Essbase Functions Supported by Calculation Manager

The following Oracle Essbase functions are supported by Calculation Manager in Block Storage applications.

Table 9-3 Essbase Functions Supported by Calculation Manager

@ABS	@ISANCEST	@MOVSUMX
@ACCUM	@ISATTRIBUTE	@NAME
@ALLANCESTORS	@ISCHILD	@NEXT
@ALIAS	@ISDESC	@NEXTS
@ALLOCATE	@ISGEN	@NEXTSIBLING
@ANCEST	@ISIANCEST	@NOTEQUAL
@ANCESTORS	@ISIBLINGS	@NPV
@ANCESTVAL	@ISICHILD	@PARENT
@ATTRIBUTE	@ISIDESC	@PARENTVAL
@ATTRIBUTEBVAL	@ISIPARENT	@POWER
@ATTRIBUTESVAL	@ISISIBLING	@PREVSIBLING
@ATTRIBUTEVAL	@ISLEV	@PRIOR
@AVG	@ISMBR	@PRIORS
@AVGRANGE	@ISMBRUDA	@PTD
@BETWEEN	@ISMBRWITHATTR	@RANGE
@CALCMODE	@ISPARENT	@RANGEFIRSTVAL
@CHILDREN	@ISRANGENONEMPTY	@RANGELASTVAL
@COMPOUND	@ISSAMEGEN	@RANK
@COMPOUNDGROWTH	@ISSAMELEV	@RDESCENDANTS
@CONCATENATE	@ISSIBLING	@RELATIVE
@CORRELATION	@ISUDA	@RELXRANGE
@COUNT	@LANCESTORS	@REMAINDER
@CREATEBLOCK	@LDESCENDANTS	@REMOVE
@CURGEN	@LEV	@RETURN
@CURLEV	@LEVMBRS	@ROUND
@CURRMBR	@LIKE	@RSIBLINGS
@CURRMBRRANGE	@LIST	@SANCESTVAL
@DATEDIFF	@LN	@SHARE
@DATEPART	@LOG	@SHIFT
@DATEROLL	@LOG10	@SHIFTMINUS
@DECLINE	@LSIBLINGS	@SHIFTPLUS
@DESCENDANTS	@MATCH	@SHIFTSIBLING
@DISCOUNT	@MAX	@SIBLINGS
@ENUMVALUE	@MAXRANGE	@SLN
@EQUAL	@MAXS	@SPARENTVAL
@EXP	@MAXSRANGE	@SPLINE
@EXPAND	@MBRCOMPARE	@STDEV
@FACTORIAL	@MBRPARENT	@STDEVP

Table 9-3 (Cont.) Essbase Functions Supported by Calculation Manager

@FORMATDATE	@MDALLOCATE	@STDEVRANGE
@GEN	@MDANCESTVAL	@SUBSTRING
@GENMBRS	@MDPARENTVAL	@SUM
@GROWTH	@MDSHIFT	@SUMRANGE
@IALLANCESTORS	@MEDIAN	@SYD
@IANCESTORS	@MEMBER	@TODATE
@ICHILDREN	@MEMBERAT	@TODATEEX
@IDESCENDANTS	@MERGE	@TODAY
@IALLANCESTORS	@MIN	@TREND
@ILDESCENDANTS	@MINRANGE	@TRUNCATE
@ILSIBLINGS	@MINS	@UDA
@INT	@MINSRANGE	@VAR
@INTEREST	@MOD	@VARPER
@INTERSECT	@MODE	@VARIANCE
@IRDESCENDANTS	@MOVAVG	@VARIANCEP
@IRR	@MOVMAX	@WITHATTR
@IRREX	@MOVMED	@XRANGE
@IRSIBLINGS	@MOVMIN	@XREF
@ISACCTYPE	@MOVSUM	@XWRITE

Working with Custom Functions

Use custom functions to perform tasks such as copying and exporting data, removing and adding single or double quotes to a text string, comparing two text strings, and converting dates to other formats.

Related Topics

- About Custom Functions
- Using a Custom Function with a Year Parameter
- @CalcMgrExcel Custom Functions with Date Parameters
- Bitwise Functions
- Counter Functions
- Date/Time Functions
- Financial Functions
- Log Functions
- Math Functions
- Statistical Functions
- String Functions

About Custom Functions

You access custom functions from the function selector in Calculation Manager condition, script, and formula components.

Custom functions allow you to perform tasks such as copying and exporting data, removing and adding single or double quotes to a text string, comparing two strings, converting a date to the `YYYYMMDD` format, among other tasks.

You can use custom functions in Planning business rule components.

Using a Custom Function with a Year Parameter

In some custom functions, the parameter may have a drop down with a selection of *year*. If you have a dimension or member in your application named *year*, using the custom function with the selection of *year* will not validate. This could be an issue for any parameter selection (not just *year*) that is also a dimension or member name.

To work around this issue, after selecting *year* from the parameter's drop down, in the rule, add `@name( )` around it, so that it shows as: `@name(year)`.

@CalcMgrExcel Custom Functions with Date Parameters

Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. You can use these functions to convert dates from a `YYYYMMDD` format (serial format) to an Excel format:

- `@CalcMgrExcelDATE` returns the serial number of a particular date.

  The following example converts `20181214` (`YYYYMMDD` format) to an Excel date

  ```
  @CalcMgrExcelDATE(20181214)
  ```

- `@CalcMgrDateToExcel` converts a single date in `YYYYMMDD` format to an Excel date
- `@CalcMgrDatesToExcel` converts multiple dates in `YYYYMMDD` format to Excel dates

  In the following example:

  ```
  @CalcMgrDatesToExcel(@LIST("Jan"->"Date_123", "Feb"->Date_123)
  ```

  `"Jan"->"Date_123"` will display as `01/31/19` and `"Feb"->Date_123` will display as `02/31/19`

Bitwise Functions

Related Topics

- @CalcMgrBitAnd
- @CalcMgrBitOR
- @CalcMgrBitExOR
- @CalcMgrBitExBoolOR
- @CalcMgrBitCompliment
- @CalcMgrBitShiftLeft
- @CalcMgrBitShiftRight
- @CalcMgrBitUnsignedShiftRight

@CalcMgrBitAnd

Purpose:

Performs a bitwise `AND` operation, which compares each bit of the first operand to the corresponding bit of the second operand. If both bits are 1, the corresponding result bit is set to 1; otherwise, the corresponding result bit is set to 0.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.BitwiseFunctions.AND(double,double)

CDF Spec: @CalcMgrBitAnd(number1,number2)
```

@CalcMgrBitOR

Purpose:

Performs a bitwise `OR` operation, which compares each bit of the first operand to the corresponding bit of the second operand. If either bit is 1, the corresponding result bit is set to 1; otherwise, the corresponding result bit is set to 0.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.BitwiseFunctions.OR(double,double)

CDF Spec: @CalcMgrBitOR(number1,number2
```

@CalcMgrBitExOR

Purpose:

Performs an exclusive bitwise `OR` operation, which compares each bit of the first operand to the corresponding bit of the second operand. If either bit is 1, the corresponding result bit is set to 1; otherwise, the corresponding result bit is set to 0.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.BitwiseFunctions.EXCLUSIVEOR(double,double)

CDF Spec: @CalcMgrBitExOR(number1,number2)
```

@CalcMgrBitExBoolOR

Purpose:

Performs an exclusive bitwise `OR` operation, which compares each bit of the first operand to the corresponding bit of the second operand. If either bit is 1, the corresponding result bit is set to 1; otherwise, the corresponding result bit is set to 0.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.BitwiseFunctions.EXCLUSIVEOR(double,double)

CDF Spec: @CalcMgrBitExOR(number1,number2)
```

@CalcMgrBitCompliment

Purpose:

Performs a unary bitwise complement, which reverses each bit.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.BitwiseFunctions.UNARYCOMPLIMENT(double)

CDF Spec: @CalcMgrBitCompliment(number1)
```

@CalcMgrBitShiftLeft

Purpose:

Performs a signed left shift.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.BitwiseFunctions.SIGNEDSHIFTLEFT(double,dou
ble)

CDF Spec: @CalcMgrBitShiftLeft(number1,number2)
```

@CalcMgrBitShiftRight

Purpose:

Performs a signed right shift.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.BitwiseFunctions.SIGNEDSHIFTRIGHT(double,do
uble)

CDF Spec: @CalcMgrBitShiftRight(number1,number2)
```

@CalcMgrBitUnsignedShiftRight

Purpose:

Performs an unsigned right shift.

Syntax:

```
com.hyperion.calcmgr.common.excel.cdf.BitwiseFunctions.UNSIGNEDSHIFTRIGHT(double,
double)

@CalcMgrBitUnsignedShiftRight(number1,number2)
```

Counter Functions

Related Topics

- @CalcMgrCounterAddNumber

- @CalcMgrCounterAddText
- @CalcMgrCounterClear
- @CalcMgrCounterClearAll
- @CalcMgrCounterClearKey
- @CalcMgrCounterDecrement
- @CalcMgrCounterDecrementKey
- @CalcMgrCounterGetKeyNumber
- @CalcMgrCounterGetKeyText
- @CalcMgrCounterGetNumber
- @CalcMgrCounterGetText
- @CalcMgrCounterIncrement
- @CalcMgrCounterIncrementKey
- @CalcMgrCounterUpdate
- @CalcMgrCounterUpdateNumber
- @CalcMgrCounterUpdateNumberText
- @CalcMgrCounterUpdateText

@CalcMgrCounterAddNumber

Purpose:

Adds a number to the counter and returns the key

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.CounterFunctions.addNumber(double)

CDF Spec: @CalcMgrCounterAddNumber(number)
```

@CalcMgrCounterAddText

Purpose:

Adds a text string to the counter and returns the key

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.CounterFunctions.addText(String)

CDF Spec: @CalcMgrCounterAddText(text)
```

@CalcMgrCounterClear

Purpose:

Clears the counter specified by the key

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.CounterFunctions.clear(double)

CDF Spec: @CalcMgrCounterClear(key)
```

@CalcMgrCounterClearAll

Purpose:

Removes all keys and values from the counter

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.CounterFunctions.clearAll()
```

```
CDF Spec: @CalcMgrCounterClearAll()
```

@CalcMgrCounterClearKey

Purpose:

Removes the value from the counter associated with the key

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.CounterFunctions.clearKey(String)
```

```
CDF Spec: @CalcMgrCounterClearKey(key)
```

@CalcMgrCounterDecrement

Purpose:

Decrements the value in the counter based on the key. If the key is not found, a value of zero is set for the key

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.CounterFunctions.decrement(double)
```

```
CDF Spec: @CalcMgrCounterDecrement(key)
```

@CalcMgrCounterDecrementKey

Purpose:

Decrements the value in the counter based on the key. If the key is not found, a value of zero is set for the key

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.CounterFunctions.decrementKey(String)
```

```
CDF Spec: @CalcMgrCounterDecrementKey(key)
```

@CalcMgrCounterGetKeyNumber

Purpose:

Returns the text found in the counter based on the key. If the key is not found, missing value is returned.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CounterFunctions.getKeyNumber(String,double)

CDF Spec: @CalcMgrCounterGetKeyNumber(key, missing_value)
```

@CalcMgrCounterGetKeyText

Purpose:

Returns the text found in the counter based on the key. If the key is not found, missing value is returned.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CounterFunctions.getKeyText(String,String)

CDF Spec: @CalcMgrCounterGetKeyText(key,missing_value)
```

@CalcMgrCounterGetNumber

Purpose:

Returns the number from the counter specified by the key. If the key is not found or the value is not a number, missing value is returned.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CounterFunctions.getNumber(double,double)

CDF Spec: @CalcMgrCounterGetNumber(key,missingValue)
```

@CalcMgrCounterGetText

Purpose:

Returns the text found in the counter based on the key. If the key is not found, missing value is returned.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CounterFunctions.getText(double,String)

CDF Spec: @CalcMgrCounterGetText(key,missing_value)
```

@CalcMgrCounterIncrement

Purpose:

Increment the value in the counter specified by the key

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.CounterFunctions.increment(double)

CDF Spec: @CalcMgrCounterIncrement(key)
```

@CalcMgrCounterIncrementKey

Purpose:

Increments the value in the counter based on the key. If the key is not found, a value of zero is set for the key.

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.CounterFunctions.incrementKey(String)

CDF Spec: @CalcMgrCounterIncrementKey(key)
```

@CalcMgrCounterUpdate

Purpose:

Sets the number in the counter with the key specified

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CounterFunctions.update(double,double)

CDF Spec: @CalcMgrCounterUpdate(key, number)
```

@CalcMgrCounterUpdateNumber

Purpose:

Updates the number in the counter with the key specified

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CounterFunctions.updateNumber(String,double)

CDF Spec: @CalcMgrCounterUpdateNumber(key, number)
```

@CalcMgrCounterUpdateNumberText

Purpose:

Updates the number in the counter with the key specified

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CounterFunctions.updateNumberText(double,String)

CDF Spec: @CalcMgrCounterUpdateNumberText(key,number)
```

@CalcMgrCounterUpdateText

Purpose:

Updates the text in the counter with the key specified

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CounterFunctions.updateText(String,String)

CDF Spec: @CalcMgrCounterUpdateText(key, text)
```

Date/Time Functions

Related Topics

- @CalcMgrAddDate
- @CalcMgrAddDatePart
- @CalcMgrAddDays
- @CalcMgrAddMonths
- @CalcMgrAddWeeks
- @CalcMgrAddYears
- @CalcMgrDateDiff
- @CalcMgrDateToExcel
- @CalcMgrDatesToExcel
- @CalcMgrDateTimeToExcel
- @CalcMgrDateTimesToExcel
- @CalcMgrDateToString
- @CalcMgrDaysBetween
- @CalcMgrDaysDiff
- @CalcMgrDiffDate
- @CalcMgrExcelADD
- @CalcMgrExcelDATE
- @CalcMgrExcelDATEDIF
- @CalcMgrExcelDAYOFYEAR
- @CalcMgrExcelDAYS360
- @CalcMgrExcelDAYSINMONTH
- @CalcMgrExcelEOMONTH
- @CalcMgrExcelHOUR
- @CalcMgrExcelMINUTE
- @CalcMgrExcelMONTH
- @CalcMgrExcelNETWORKDAYS
- @CalcMgrExcelSECOND
- @CalcMgrExcelToDate
- @CalcMgrExcelToDateTime
- @CalcMgrExcelWEEKNUM
- @CalcMgrExcelWEEKDAY
- @CalcMgrExcelWORKDAY

- @CalcMgrExcelYEAR
- @CalcMgrExcelYEARFRAC
- @CalcMgrGetCurrentDate
- @CalcMgrGetCurrentDateTZ
- @CalcMgrGetCurrentDateTime
- @CalcMgrGetCurrentDateTimeTZ
- @CalcMgrGetCustomDate
- @CalcMgrGetCustomDateTime
- @CalcMgrGetDatePart
- @CalcMgrGetDateTimePart
- @CalcMgrGetDay
- @CalcMgrGetDayOfYear
- @CalcMgrGetFormattedDate
- @CalcMgrGetMaxDaysInMonth
- @CalcMgrGetMonth
- @CalcMgrGetStringFormattedDateTime
- @CalcMgrGetWeekOfMonth
- @CalcMgrGetWeekOfYear
- @CalcMgrGetYear
- @CalcMgrIsLeapYear
- @CalcMgrMonthsBetween
- @CalcMgrMonthsDiff
- @CalcMgrRollDate
- @CalcMgrRollDay
- @CalcMgrRollMonth
- @CalcMgrRollYear
- @CalcMgrWeeksBetween
- @CalcMgrWeeksDiff
- @CalcMgrYearsBetween
- @CalcMgrYearsDiff

@CalcMgrAddDate

Purpose:

Adds a specified number of years, months, and days to a date that is in the `YYYYMMDD` format

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CalendarFunctions.addDate(int,int,int,int)

CDF Spec: @CalcMgrAddDate(date, years, months, days)
```

@CalcMgrAddDatePart

Purpose:

Adds a specified number of years/months/days/weeks to the date that is in the YYYYMMDD format. The "date_part" can be one of the following: "day", "month", "week", "year"

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CalendarFunctions.addDatePart(int,String,int)

CDF Spec: @CalcMgrAddDatePart(date,date_part, amountToAdd)
```

@CalcMgrAddDays

Purpose:

Adds a specified number of days to a date that is in the YYYYMMDD format

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.CalendarFunctions.addDays(int,int)

CDF Spec: @CalcMgrAddDays(date, daysToAdd)
```

@CalcMgrAddMonths

Purpose:

Adds a specified number of months to the date that is in the YYYYMMDD format

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.CalendarFunctions.addMonths(int,int)

CDF Spec: @CalcMgrAddMonths(date, monthsToAdd)
```

@CalcMgrAddWeeks

Purpose:

Adds a specified number of weeks to a date that is in the YYYYMMDD format

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.CalendarFunctions.addWeeks(int,int)

CDF Spec: @CalcMgrAddWeeks(date, weeksToAdd)
```

@CalcMgrAddYears

Purpose:

Adds a specified number of years to the date that is in the YYYYMMDD format

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.CalendarFunctions.addYears(int,int)
```

```
CDF Spec: @CalcMgrAddYears(date, yearsToAdd)
```

@CalcMgrDateDiff

Purpose:

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

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CalendarFunctions.dateDiff(int,int,String)

CDF Spec: @CalcMgrDateDiff(fromDate,toDate,datePart)
```

> **Note**
>
> `@CalcMgrDateDiff` returns only positive numbers. If you want to return a negative number if it applies, then use @CalcMgrDiffDate.

@CalcMgrDateToExcel

Purpose:

Converts a single date in `YYYYMMDD` format to an Excel date

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelDateTimeFunctions.DATE_TOEXCEL(double)
CDF Spec: @CalcMgrDateToExcel(date)
```

@CalcMgrDatesToExcel

Purpose:

Converts multiple dates in `YYYYMMDD` format to Excel dates

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelDateTimeFunctions.DATES_TOEXCEL(doubl
e[])

CDF Spec: @CalcMgrDatesToExcel(dates)
```

@CalcMgrDateTimeToExcel

Purpose:

Converts a single date in `YYYYMMDDHHMMSS` format to an Excel date

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelDateTimeFunctions.DATETIME_TOEXCEL(dou
ble)
CDF Spec: @CalcMgrDateTimeToExcel(date)
```

@CalcMgrDateTimesToExcel

Purpose:

Converts multiple dates in `YYYYMMDDHHMMSS` format to Excel dates

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelDateTimeFunctions.DATETIMES_TOEXCEL(do
uble[])

CDF Spec: CalcMgrDateTimesToExcel(dates)
```

@CalcMgrDateToString

Purpose:

Returns the date in the `YYYYMMDD` format, as a string using the format given. For the format, see `SimpleDateFormat` in Java documentation

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CalendarFunctions.dateToString(int,String)

CDF Spec: @CalcMgrDateToString(date, format)
```

@CalcMgrDaysBetween

Purpose:

Returns the days between two dates that are in the `YYYYMMDD` format

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CalendarFunctions.daysBetween(int,int)
CDF Spec: @CalcMgrDaysBetween(fromDate,toDate)
```

> **Note**
>
> `@CalcMgrDaysBetween` returns only positive numbers. If you want to return a negative number if it applies, then use @CalcMgrDaysDiff.

@CalcMgrDaysDiff

Purpose:

Returns the days between two dates that are in the `YYYYMMDD` format

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.CalendarFunctions.daysDiff(int,int)
CDF Spec: @CalcMgrDaysDiff(fromDate,toDate)
```

> **Note**
>
> When using `@CalcMgrDaysDiff`, if the first date is later than then second date in the function, then a negative number is returned. If the first date is before the second date in the function, then a positive number is returned. if you want to return only positive numbers, then use @CalcMgrDaysBetween.

@CalcMgrDiffDate

Purpose:

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

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CalendarFunctions.diffDate(int,int,String)

CDF Spec: @CalcMgrDiffDate(fromDate,toDate,datePart)
```

> **Note**
>
> When using `@CalcMgrDiffDate`, if the first date is later than the second date in the function, then a negative number is returned. If the first date is before the second date in the function, then a positive number is returned. If you want to return only positive numbers, then use @CalcMgrDateDiff.

@CalcMgrExcelADD

Purpose:

Adds an amount to date

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelDateTimeFunctions.ADD(double,double,St
ring)

CDF Spec: @CalcMgrExcelADD(date, amount, what)
```

@CalcMgrExcelDATE

Purpose:

Returns the serial number of a particular date

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelDateTimeFunctions.DATE(double,double,d
ouble)

CDF Spec: @CalcMgrExcelDATE(year,month,day)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters

@CalcMgrExcelDATEDIF

Purpose:

Calculates the number of days, months, or years between two dates.

Useful in formulas where you need to calculate an age

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelDateTimeFunctions.DATEDIF(double,doubl
e,String)

CDF Spec: @CalcMgrExcelDATEDIF(start_date, end_date,unit)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelDAYOFYEAR

Purpose:

Converts a serial number to a day of the year

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelDateTimeFunctions.DAYOFYEAR(double)

CDF Spec: @CalcMgrExcelDAYOFYEAR(Date)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelDAYS360

Purpose:

Calculates the number of days between two dates based on a 360-day year

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelDateTimeFunctions.DAYS360(double,double,boolean)

CDF Spec: @CalcMgrExcelDAYS360(start_date, end_date, method)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelDAYSINMONTH

Purpose:

Converts a serial number to days in the month

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelDateTimeFunctions.DAYSINMONTH(double)

CDF Spec: @CalcMgrExcelDAYSINMONTH(date)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelEOMONTH

Purpose:

Returns the serial number of the last day of the month before or after a specified number of months

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelDateTimeFunctions.EOMONTH(double,double)

CDF Spec: @CalcMgrExcelEOMONTH(dateValue, adjustmentMonths)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelHOUR

Purpose:

Converts a serial number hour of the day

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelDateTimeFunctions.HOUR(double)

CDF Spec: @CalcMgrExcelHOUR(date)
```

@CalcMgrExcelMINUTE

Purpose:

Converts a serial number to a minute

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelDateTimeFunctions.MINUTE(double)

CDF Spec: @CalcMgrExcelMINUTE(date)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelMONTH

Purpose:

Converts a serial number to a month

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelDateTimeFunctions.MONTH(double)

CDF Spec: @CalcMgrExcelMONTH(Date)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelNETWORKDAYS

Purpose:

Returns the number of whole workdays between two dates

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelDateTimeFunctions.NETWORKDAYS(double,double,double[])

CDF Spec: @CalcMgrExcelNETWORKDAYS(startDate, endDate, holidays)
```

> **Note**
>
> If you are passing a date in a Planning date format, you must convert the Planning date to an Excel date using @CalcMgrDateToExcel.

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelSECOND

Purpose:

Converts a serial number to second

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelDateTimeFunctions.SECOND(double)

CDF Spec: @CalcMgrExcelSECOND(date)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelToDate

Purpose:

Converts an Excel date to `YYYYMMDD` format.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelDateTimeFunctions.EXCEL_TODATE(double)
CDF Spec: @CalcMgrExcelToDate(excel_date)
```

@CalcMgrExcelToDateTime

Purpose:

Converts an Excel date to `YYYYMMDDHHMMSS` format.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelDateTimeFunctions.EXCEL_TODATETIME(dou
ble)
CDF Spec: @CalcMgrExcelToDateTime(excel_date)
```

@CalcMgrExcelWEEKNUM

Purpose:

Returns the week number of a specific date. For example, the week containing January 1 is the first week of the year, and is numbered week 1.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelDateTimeFunctions.WEEKNUM(double,doubl
e)

CDF Spec: @CalcMgrExcelWEEKNUM(date, method)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelWEEKDAY

Purpose:

Returns the day of the week corresponding to a date. The day is given as an integer, ranging from 1 (Sunday) to 7 (Saturday), by default.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelDateTimeFunctions.WEEKDAY(double,doubl
e)

CDF Spec: @CalcMgrExcelWEEKDAY(serial_number, return_type)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelWORKDAY

Purpose:

Returns the serial number of the date before or after a specified number of workdays

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelDateTimeFunctions.WORKDAY(double,doubl
e,double[])

CDF Spec: @CalcMgrExcelWORKDAY(startDate, days, holidays)
```

> **Note**
>
> If you are passing a date in a Planning date format, you must convert the Planning date to an Excel date using @CalcMgrDateToExcel.

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelYEAR

Purpose:

Converts a serial number to year

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelDateTimeFunctions.YEAR(double)

CDF Spec: @CalcMgrExcelYEAR(date)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelYEARFRAC

Purpose:

Returns the year fraction representing the number of whole days between `start_date` and `end_date`

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelDateTimeFunctions.YEARFRAC(double,doub
le,double)

CDF Spec: @CalcMgrExcelYEARFRAC(startDate, endDate, basis)
```

> **Note**
>
> If you are passing a date in a Planning date format, you must convert the Planning date to an Excel date using @CalcMgrDateToExcel.

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrGetCurrentDate

Purpose:

Returns the current date in the `YYYYMMDD` format (for example: 20140101)

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.CalendarFunctions.getCurrentDate(int)

CDF Spec: @CalcMgrGetCurrentDate()
```

@CalcMgrGetCurrentDateTZ

Purpose:

Returns the current date in the time zone provided in `YYYYMMDD` format (for example: 20140101)

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CalendarFunctions.getCurrentDate(String)
```

```
CDF Spec: @CalcMgrGetCurrentDateTZ(timeZone)
```

@CalcMgrGetCurrentDateTime

Purpose:

Returns the current date and time in the `YYYYMMDDHHMMSS` format. For example: 20140101143001 (Year_Month_Day_Hour_Minute_Second)

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CalendarFunctions.getCurrentDateTime()

CDF Spec: @CalcMgrGetCurrentDateTime()
```

@CalcMgrGetCurrentDateTimeTZ

Purpose:

Returns the current date and time in the time zone provided in the `YYYYMMDDHHMMSS` format. For example: 20140101143001 (Year_Month_Day_Hour_Minute_Second)

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CalendarFunctions.getCurrentDateTime(String)

CDF Spec: @CalcMgrGetCurrentDateTimeTZ(timeZone)
```

@CalcMgrGetCustomDate

Purpose:

Returns a custom date in the `YYYYMMDD` format.

For example, 20140101

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CalendarFunctions.getDate(double,double,double)

CDF Spec: @CalcMgrGetCustomDate(year, month, day)
```

@CalcMgrGetCustomDateTime

Purpose:

Returns the custom date and time in the `YYYYMMDDHHMMSS` formatFor example: 20140101143001 (Year_Month_Day_Hour_Minute_Second)

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CalendarFunctions.getDateTime(double,double,doubl
e,double,double,double):

CDF Spec: @CalcMgrGetCustomDateTime(year, month, day, hour, min, sec)
```

@CalcMgrGetDatePart

Purpose:

Returns the `Year/Month/DayOfMonth/WeekOfYear/WeekOfMonth/DayOfYear` as a number from a date that is in the `YYYYMMDD` format. "date_part_ex" can be: "dayofmonth", "dayofyear", "month", "weekofmonth", "weekofyear", or "year"

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CalendarFunctions.datePart(Double,String)

CDF Spec: @CalcMgrGetDatePart(date, date_part_ex)
```

@CalcMgrGetDateTimePart

Purpose:

Returns the `Year/Month/DayOfMonth/WeekOfYear/WeekOfMonth/DayOfYear/Hour/Minute/Seconds` as a number from date.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CalendarFunctions.dateTimePart(double,String)

CDF Spec: @CalcMgrGetDateTimePart(date, date_part_ex)
```

@CalcMgrGetDay

Purpose:

Returns the day from a date that is in the `YYYYMMDD` format

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.CalendarFunctions.getDay(int)

CDF Spec: @CalcMgrGetDay(date)
```

@CalcMgrGetDayOfYear

Purpose:

Returns the day of year (1-366) from a date that is in the `YYYYMMDD` format.

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.CalendarFunctions.getDayOfYear(int)

CDF Spec: @CalcmgrGetDayOfYear(date)
```

@CalcMgrGetFormattedDate

Purpose:

Converts the date to the `YYYYMMDD` format. For example, `@CalcMgrGetFormattedDate(12302014, "mmddyyyy")` returns 20141230

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CalendarFunctions.getFormattedDate(int,String)

CDF Spec: @CalcMgrGetFormattedDate(date, format)
```

@CalcMgrGetMaxDaysInMonth

Purpose:

Returns the maximum days in the month of date that is in the `YYYYMMDD` format

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CalendarFunctions.getActualMaximumDays(int)

CDF Spec: @CalcMgrGetMaxDaysInMonth(date)
```

@CalcMgrGetMonth

Purpose:

Returns the month from a date that is in the `YYYYMMDD` format

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.CalendarFunctions.getMonth(int)

CDF Spec: @CalcMgrGetMonth(date)
```

@CalcMgrGetStringFormattedDateTime

Purpose:

Converts the date defined by format to date in the `YYYYMMddHHmmss` format.

For example: `@CalcMgrGetFormattedDate(12302014, "MMddyyyyHHmmss")` returns 201412301430.

For more information, see "SimpleDateFormat " in the Java docs.

Possible values for format: `mmddyyyyHHmmss`, `ddmmyyyyHHmmss`, `yyyyddmmHHmmss`

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CalendarFunctions.getStringFormattedDateTime(Stri
ng,String)
CDF Spec: @CalcMgrGetStringFormattedDateTime(date, format)
```

@CalcMgrGetWeekOfMonth

Returns the week of month from a date that is in the `YYYYMMDD` format

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.CalendarFunctions.getWeekOfMonth(int)

CDF Spec: @CalcMgrGetWeekOfMonth(date)
```

@CalcMgrGetWeekOfYear

Purpose:

Returns the week of the year from a date that is in the YYYYMMDD format

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.CalendarFunctions.getWeekOfYear(int)

CDF Spec: @CalcMgrGetWeekOfYear(date)
```

@CalcMgrGetYear

Purpose:

Returns the year from a date that is in the YYYYMMDD format "date_part_ex" must be: "dayofmonth", "dayofyear", "month", "weekofmonth", "weekofyear", or "year"

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.CalendarFunctions.getYear(int)

CDF Spec: @CalcMgrGetYear(date)
```

@CalcMgrIsLeapYear

Purpose:

Determines whether the given date is a leap year. The date must be in YYYYMMDD or YYYY format (for example: 20140101 or 2014)

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.CalendarFunctions.isLeapYear(int)
CDF Spec: @CalcMgrIsLeapYear(date)
```

@CalcMgrMonthsBetween

Purpose:

Returns the months between two dates that are in the YYYYMMDD format

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CalendarFunctions.monthsBetween(int,int)
CDF Spec: @CalcMgrMonthsBetween(fromDate,toDate)
```

> **Note**
>
> @CalcMgrMonthsBetween returns only positive numbers. If you want to return a negative number if it applies, then use @CalcMgrMonthsDiff.

@CalcMgrMonthsDiff

Purpose:

Returns the months between two dates that are in the YYYYMMDD format

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.CalendarFunctions.monthsDiff(int,int)
CDF Spec: @CalcMgrMonthsDiff(fromDate,toDate)
```

> **Note**
>
> When using @CalcMonthsDiff, if the first date is later than the second date in the function, then a negative number is returned. if the first date is before the second date in the function, then a positive number is returned. If you want to return only positive numbers, then use @CalcMgrMonthsBetween.

@CalcMgrRollDate

Purpose:

Adds or subtracts (up or down) a single unit of time on the given date field without changing larger fields.

For example, @CalcMgrRollDate(19960131,"month",@_true) results in the date of 19960229. @CalcMgrRollDate(19960131,"day",@_true) results in the date of 19960101.

Possible values of date_part are: day, month, week and year.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CalendarFunctions.rollDate(int,String,boolean)

CDF Spec: @CalcMgrRollDate(date,date_part,up)
```

@CalcMgrRollDay

Purpose:

Roll the day up or down to the date which is in the YYYYMMDD format

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CalendarFunctions.rollDay(int,boolean)

CDF Spec: @CalcMgrRollDay(date,up)
```

@CalcMgrRollMonth

Purpose:

Roll the month up or down to the date which is in the YYYYMMDD format.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CalendarFunctions.rollMonth(int,boolean)

CDF Spec: @CalcMgrRollMonth(date,up)
```

@CalcMgrRollYear

Purpose:

Roll the year up or down to the date which is in the `YYYYMMDD` format.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CalendarFunctions.rollYear(int,boolean)

CDF Spec: @CalcMgrRollYear(date,up)
```

@CalcMgrWeeksBetween

Purpose:

Returns the weeks between two dates that are in the `YYYYMMDD` format

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CalendarFunctions.weeksBetween(int,int)
CDF Spec: @CalcMgrWeeksBetween(fromDate,toDate)
```

> **Note**
>
> `@CalcMgrWeeksBetween` returns only positive numbers. If you want to return a negative number if it applies, then use @CalcMgrWeeksDiff.

@CalcMgrWeeksDiff

Purpose:

Returns the weeks between two dates that are in the `YYYYMMDD` format

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.CalendarFunctions.weeksDiff(int,int)
CDF Spec: @CalcMgrWeeksDiff(fromDate,toDate)
```

> **Note**
>
> When using `@CalcMgrWeeksDiff`, if the first date is later than the second date in the function, then a negative number is returned. If the first date is before the second date in the function, then a positive number is returned. If you want to return only positive numbers, then use @CalcMgrWeeksBetween.

@CalcMgrYearsBetween

Purpose:

Returns the years between two dates that are in the `YYYYMMDD` format

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.CalendarFunctions.yearsBetween(int,int)

CDF Spec: @CalcMgrYearsBetween(fromDate,toDate)
```

> **Note**
>
> `@CalcMgrYearsBetween` returns only positive numbers. If you want to return a negative number if it applies, then use @CalcMgrYearsDiff.

@CalcMgrYearsDiff

Purpose:

Returns the years between two dates that are in the `YYYYMMDD` format

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.CalendarFunctions.yearsDiff(int,int)

CDF Spec: @CalcMgrYearsDiff(fromDate,toDate)
```

> **Note**
>
> When using `@CalcMgrYearsDiff`, if the first date is later than the second date in the function, then a negative number is returned. If the first date is before the second date in the function, then a positive number is returned. If you want to return only positive numbers, then use @CalcMgrYearsBetween.

Financial Functions

Related Topics

- @CalcMgrExcelACCRINT
- @CalcMgrExcelACCRINTM
- @CalcMgrExcelAMORDEGRC
- @CalcMgrExcelAMORLINC
- @CalcMgrExcelCOUPDAYBS
- @CalcMgrExcelCOUPDAYS
- @CalcMgrExcelCOUPDAYSNC
- @CalcMgrExcelCOUPNCD

- @CalcMgrExcelCOUPNUM
- @CalcMgrExcelCOUPPCD
- @CalcMgrExcelCUMIPMT
- @CalcMgrExcelCUMPRINC
- @CalcMgrExcelDB
- @CalcMgrExcelDDB
- @CalcMgrExcelDISC
- @CalcMgrExcelDOLLARDE
- @CalcMgrExcelDOLLARFR
- @CalcMgrExcelDURATION
- @CalcMgrExcelEFFECT
- @CalcMgrExcelFV
- @CalcMgrExcelFVSCHEDULE
- @CalcMgrExcelMDURATION
- @CalcMgrExcelINTRATE
- @CalcMgrExcelPMT
- @CalcMgrExcelIRR
- @CalcMgrExcelISPMT
- @CalcMgrExcelMIRR
- @CalcMgrExcelNPER
- @CalcMgrExcelNPV
- @CalcMgrExcelPPMT
- @CalcMgrExcelPRICE
- @CalcMgrExcelPRICEDISC
- @CalcMgrExcelPRICEMAT
- @CalcMgrExcelPV
- @CalcMgrExcelRATE
- @CalcMgrExcelRECEIVED
- @CalcMgrExcelSLN
- @CalcMgrExcelSYD
- @CalcMgrExcelTBILLEQ
- @CalcMgrExcelTBILLPRICE
- @CalcMgrExcelTBILLYIELD
- @CalcMgrExcelXIRR
- @CalcMgrExcelXNPV
- @CalcMgrExcelYIELD
- @CalcMgrExcelYIELDDISC
- @CalcMgrExcelYIELDMAT

@CalcMgrExcelACCRINT

Purpose:

Returns the accrued interest for a security that pays periodic interest

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.ACCRINT(double,doub
le,double,double,double,double,double,boolean)

CDF Spec: @CalcMgrExcelACCRINT(issue, firstinterest, settlement, rate, par,
frequency, basis, method)
```

@CalcMgrExcelACCRINTM

Purpose:

Returns the accrued interest for a security that pays interest at maturity

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.ACCRINTM(issue,
settlement, rate, par, basis)

CDF Spec:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.ACCRINTM(double,dou
ble,double,double,double)
```

@CalcMgrExcelAMORDEGRC

Purpose:

Returns the depreciation for each accounting period by using a depreciation coefficient

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.AMORDEGRC(double,do
uble,double,double,double,double,double)

CDF Spec: @CalcMgrExcelAMORDEGRC(cost, purchased, firstPeriod, salvage, period,
rate, basis)
```

@CalcMgrExcelAMORLINC

Purpose:

Returns the depreciation for each accounting period

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.AMORLINC(double,dou
ble,double,double,double,double,double)
```

```
CDF Spec: @CalcMgrExcelAMORLINC(cost, date_purchased, first_period, salvage,
period, rate, basis)
```

@CalcMgrExcelCOUPDAYBS

Purpose:

Returns the number of days from the beginning of the coupon period to the settlement date

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.COUPDAYBS(double,do
uble,double,double)

CDF Spec: @CalcMgrExcelCOUPDAYBS(settlement, maturity, frequency, basis)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelCOUPDAYS

Purpose:

Returns the number of days in the coupon period that contains the settlement date

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.COUPDAYS(double,dou
ble,double,double)

CDF Spec: @CalcMgrExcelCOUPDAYS(settlement, maturity, frequency, basis)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelCOUPDAYSNC

Purpose:

Returns the number of days from the settlement date to the next coupon date

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.COUPDAYSNC(double,d
ouble,double,double)

CDF Spec: @CalcMgrExcelCOUPDAYSNC(settlement, maturity, frequency, basis)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelCOUPNCD

Purpose:

Returns a number that represents the next coupon date after the settlement date

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.COUPNCD(double,doub
le,double,double)

CDF Spec: @CalcMgrExcelCOUPNCD(settlement, maturity, frequency, basis)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelCOUPNUM

Purpose:

Returns the number of coupons payable between the settlement date and maturity date, rounded up to the nearest whole coupon

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.COUPNUM(double,doub
le,double,double)

CDF Spec: @CalcMgrExcelCOUPNUM(settlement, maturity, frequency, basis)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelCOUPPCD

Purpose:

Returns a number that represents the previous coupon date before the settlement date

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.COUPPCD(double,doub
le,double,double)

CDF Spec: @CalcMgrExcelCOUPPCD(settlement, maturity, frequency, basis)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelCUMIPMT

Purpose:

Returns the cumulative interest paid on a loan between `start_period` and `end_period`

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.CUMIPMT(double,doub
le,double,double,double,double)

CDF Spec: @CalcMgrExcelCUMIPMT(rate, nper, pv, start_period, end_period, type)
```

@CalcMgrExcelCUMPRINC

Purpose:

Returns the cumulative principal paid on a loan between the start period and the end period

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.CUMPRINC(double,dou
ble,double,double,double,double)

CDF Spec: @CalcMgrExcelCUMPRINC(rate, per, nper, pv, fv, type)
```

@CalcMgrExcelDB

Purpose:

Returns the depreciation of an asset for a specified period using the fixed-declining balance method

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.DB(double,double,do
uble,double,double)

CDF Spec: @CalcMgrExcelDB((cost, salvage, life, period, month)
```

@CalcMgrExcelDDB

Purpose:

Returns the depreciation of an asset for a specified period using the double-declining balance method or some other method you specify

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.DDB(double,double,double,double,double)

CDF Spec: @CalcMgrExcelDDB(cost, salvage, life, period, factor)
```

@CalcMgrExcelDISC

Purpose:

Returns the discount rate for a security

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.DISC(double,double,double,double,double)

CDF Spec: @CalcMgrExcelDISC(settlement, maturity, pr, redemption, basis)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelDOLLARDE

Purpose:

Converts a dollar price expressed as an integer part and a fraction part, such as 1.02, into a dollar price expressed as a decimal number. Fractional dollar numbers are sometimes used for security prices.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.DOLLARDE(double,double)

CDF Spec: @CalcMgrExcelDOLLARDE(fractional_dollar,fraction)
```

@CalcMgrExcelDOLLARFR

Purpose:

Converts a dollar price, expressed as a decimal number, into a dollar price, expressed as a fraction

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.DOLLARFR(double,dou
ble)

CDF Spec: @CalcMgrExcelDOLLARFR(decimal_dollar, fraction)
```

@CalcMgrExcelDURATION

Purpose:

Returns the annual duration of a security with periodic interest payments**Note:** When using the `@CalcMgrExcelMDURATION` function, the calculations may not match between Calculation Manager and Excel. To make the numbers match, change the decimals to 7 and use Open Office.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.DURATION(double,dou
ble,double,double,double,double)

CDF Spec: @CalcMgrExcelDURATION(settlement, maturity, coupon, yld, frequency,
basis)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelEFFECT

Purpose:

Returns the effective annual interest rate

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.EFFECT(double,doubl
e)

CDF Spec: @CalcMgrExcelEFFECT(nominal_rate, npery)
```

@CalcMgrExcelFV

Purpose:

Returns the future value of an investment

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.FV(double,double,do
uble,double,double)

CDF Spec: @CalcMgrExcelFV(rate, nper, pmt, pv, type)
```

@CalcMgrExcelFVSCHEDULE

Purpose:

Returns the future value of an initial principal after applying a series of compound interest rates

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.FVSCHEDULE(double,d
ouble[])

CDF Spec: @CalcMgrExcelFVSCHEDULE(principal, schedule)
```

@CalcMgrExcelMDURATION

Purpose:

Returns the Macauley modified duration for a security with an assumed par value of $100**Note:** When using the `@CalcMgrExcelDURATION` function, the calculations may not match between Calculation Manager and Excel. To make the numbers match, change the decimals to 7 and use Open Office.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.MDURATION(double,do
uble,double,double,double,double)

CDF Spec: @CalcMgrExcelMDURATION(settlement,maturity,coupon,yld,frequency,basis)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelINTRATE

Purpose:

Returns the interest rate for a fully invested security

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.INTRATE(double,doub
le,double,double,double)

CDF Spec: @CalcMgrExcelINTRATE(settlement, maturity, investment, redemption,
basis)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelPMT

Purpose:

Returns the periodic payment for an annuity

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.PMT(double,double,d
ouble,double,double)

CDF Spec: @CalcMgrExcelPMT(rate, nper, pv, fv, type)
```

@CalcMgrExcelIRR

Purpose:

Returns the internal rate of return for a series of cash flows

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.IRR(double[],double
)

CDF Spec: @CalcMgrExcelIRR(values, guess)
```

@CalcMgrExcelISPMT

Purpose:

Calculates the interest paid during a specific period of an investment

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.ISPMT(double,double
,double,double)

CDF Spec: @CalcMgrExcelISPMT(rate, per, nper, pv)
```

@CalcMgrExcelMIRR

Purpose:

Returns the internal rate of return where positive and negative cash flows are financed at different rates

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.MIRR(double[],doubl
e,double)

CDF Spec: @CalcMgrExcelMIRR(values, finance_rate, reinvest_rate)
```

@CalcMgrExcelNPER

Purpose:

Returns the number of periods for an investment

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.NPER(double,double,
double,double,double)

CDF Spec: @CalcMgrExcelNPER(rate, pmt, pv, fv, type)
```

@CalcMgrExcelNPV

Purpose:

Returns the net present value of an investment based on a series of periodic cash flows and a discount rate

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.NPV(double,double[]
)

CDF Spec: @CalcMgrExcelNPV(rate, values)
```

@CalcMgrExcelPPMT

Purpose:

Returns the payment on the principal for a given period for an investment based on periodic, constant payments and a constant interest rate

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.PPMT(double,double,
double,double,double,double)

CDF Spec: @CalcMgrExcelPPMT(rate, per, nper, pv, fv, type)
```

@CalcMgrExcelPRICE

Purpose:

Returns the price per $100 face value of a security that pays periodic interest

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.PRICE(double,double
,double,double,double,double,double)

CDF Spec: @CalcMgrExcelPRICE(settlement, maturity, rate, yld, redemption,
frequency, basis)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelPRICEDISC

Purpose:

Returns the price per $100 face value of a discounted security

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.PRICEDISC(double,do
uble,double,double,double)

CDF Spec: @CalcMgrExcelPRICEDISC(settlement, maturity, discount, redemption,
basis)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelPRICEMAT

Purpose:

Returns the price per $100 face value of a security that pays interest at maturity

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.PRICEMAT(double,dou
ble,double,double,double,double)

CDF Spec: @CalcMgrExcelPRICEMAT((settlement, maturity, issue, rate, yld, basis)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelPV

Purpose:

Returns the present value of an investment

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.PV(double,double,double,double,double)

CDF Spec: @CalcMgrExcelPV(rate, nper, pmt, fv, type)
```

@CalcMgrExcelRATE

Purpose:

Returns the interest rate per period of an annuity

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.RATE(double,double,double,double,double,double)

CDF Spec: @CalcMgrExcelRATE(nper, pmt, pv, fv, type, guess)
```

@CalcMgrExcelRECEIVED

Purpose:

Returns the amount received at maturity for a fully invested security

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.RECEIVED(double,double,double,double,double)

CDF Spec: @CalcMgrExcelRECEIVED(settlement, maturity, investment, discount,
basis)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelSLN

Purpose:

Returns the straight-line depreciation of an asset for one period

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.SLN(double,double,d
ouble)

CDF Spec: @CalcMgrExcelSLN(cost, salvage, life)
```

@CalcMgrExcelSYD

Purpose:

Returns the sum-of-years' digits depreciation of an asset for a specified period

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.SYD(double,double,d
ouble,double)

CDF Spec: @CalcMgrExcelSYD(cost, salvage, life, per)
```

@CalcMgrExcelTBILLEQ

Purpose:

Returns the bond-equivalent yield for a Treasury bill

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.TBILLEQ(double,doub
le,double)

CDF Spec: @CalcMgrExcelTBILLEQ(settlement, maturity, discount)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelTBILLPRICE

Purpose:

Returns the price per $100 face value for a Treasury bill

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.TBILLPRICE(double,d
ouble,double)

CDF Spec: @CalcMgrExcelTBILLPRICE(settlement, maturity, discount)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelTBILLYIELD

Purpose:

Returns the yield for a Treasury bill

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.TBILLYIELD(double,double,double)

CDF Spec: @CalcMgrExcelTBILLYIELD(settlement, maturity, pr)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelXIRR

Purpose:

Returns the internal rate of return for a schedule of cash flows that is not necessarily periodic

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.XIRR(double[],double[],double)

CDF Spec: @CalcMgrExcelXIRR(values, dates, guess)
```

@CalcMgrExcelXNPV

Purpose:

Returns the net present value for a schedule of cash flows that is not necessarily periodic

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.XNPV(double,double[],double[])

CDF Spec: @CalcMgrExcelXNPV(rate, values, dates)
```

@CalcMgrExcelYIELD

Purpose:

Returns the yield on a security that pays periodic interest

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.YIELD(double,double
,double,double,double,double,double)

CDF Spec: @CalcMgrExcelYIELD(settlement, maturity, rate, pr, redemption,
frequency, basis)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelYIELDDISC

Purpose:

Returns the annual yield for a discounted security; for example, a Treasury bill**Note:** When using the `@CalcMgrExcelYIELDDISC` function, the calculations may not match between Calculation Manager and Excel. To make the numbers match, change the decimals to 7 and use Open Office.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.YIELDDISC(double,do
uble,double,double,double)

CDF Spec: @CalcMgrExcelYIELDDISC(settlement, maturity, pr, redemption, basis)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

@CalcMgrExcelYIELDMAT

Purpose:

Returns the annual yield of a security that pays interest at maturity**Note:** When using the `@CalcMgrExcelYIELDMAT` function, the calculations may not match between Calculation Manager and Excel. To make the numbers match, change the decimals to 7 and use Open Office.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelFinancialFunctions.YIELDMAT(double,dou
ble,double,double,double,double)

CDF Spec: @CalcMgrExcelYIELDMAT(settlement, maturity, issue, rate, pr, basis)
```

> **Note**
>
> Date parameters used in `@CalcMgrExcel` functions must be in an Excel format. See @CalcMgrExcel Custom Functions with Date Parameters.

Log Functions

Related Topics

- @CalcMgrLogMessageTrace
- @CalcMgrIsValidMember
- @CalcMgrIsValidSLMember
- @CalcMgrSLMember

@CalcMgrLogMessageTrace

Purpose:

Adds custom messages to the Log Messages tab after launching a rule in Calculation Manager.

For example, `@CalcMgrLogMessageTrace(@NAME(@CURRMBR(Product)), @NAME(@CURRMBR(Period)) );` returns the current member of Product and Period in a custom message in the Log Messages tab.

Syntax:

```
CDM Spec: @CalcMgrLogMessageTrace(Member Names)
```

> **Note**
>
> This function can only be used when the rule is launched in Calculation Manager.

@CalcMgrIsValidMember

Purpose:

Returns "true" if the input in Member Name is a valid member.

Syntax:

```
CDM Spec: @CalcMgrIsValidMember(Member Name)
```

@CalcMgrIsValidSLMember

Purpose:

Returns "true" if the member associated with the numerical Smart List value is a valid member.

The numerical Smart List value is calculated by concatenating the `HSP_ID` and the `Smartlist Value`, and by looking up the member in the Smart List alias table.

Syntax:

```
CDM Spec: @CalcMgrIsValidSLMember(Smartlist Value)
```

@CalcMgrSLMember

Purpose:

Returns the member associated with the numerical Smart List value.

The numerical Smart List value is calculated by concatenating the `HSP_ID` and the `Smartlist Value`, and by looking up the member in the Smart List alias table.

Syntax:

```
CDM Spec: @CalcMgrSLMember(Smartlist Value)
```

Math Functions

Related Topics

- @CalcMgrExcelCEILING
- @CalcMgrExcelCOMBIN
- @CalcMgrExcelEVEN
- @CalcMgrExcelFACT
- @CalcMgrExcelFLOOR
- @CalcMgrExcelGCD
- @CalcMgrExcelLCM
- @CalcMgrExcelMROUND
- @CalcMgrExcelMULTINOMIAL
- @CalcMgrExcelODD
- @CalcMgrExcelPOWER
- @CalcMgrExcelPRODUCT
- @CalcMgrExcelROUNDDOWN
- @CalcMgrExcelROUNDUP
- @CalcMgrExcelSQRT
- @CalcMgrExcelSQRTPI
- @CalcMgrExcelSUMPRODUCT
- @CalcMgrExcelSUMSQ

@CalcMgrExcelCEILING

Purpose:

Rounds a number up (away from zero) to the nearest integer or to the nearest multiple of significance

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelMathFunctions.CEILING(double,double)

CDF Spec: @CalcMgrExcelCEILING(number, significance)
```

@CalcMgrExcelCOMBIN

Purpose:

Returns the number of combinations for a given number of objects

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelMathFunctions.COMBIN(double,double)

CDF Spec: @CalcMgrExcelCOMBIN(number, number_chosen)
```

@CalcMgrExcelEVEN

Purpose:

Rounds a number up to the nearest even integer

Syntax:

```
Java Class: com.hyperion.calcmgr.common.excel.cdf.ExcelMathFunctions.EVEN(double)

CDF Spec: @CalcMgrExcelEVEN(number)
```

@CalcMgrExcelFACT

Purpose:

Returns the factorial of a number

Syntax:

```
Java Class: com.hyperion.calcmgr.common.excel.cdf.ExcelMathFunctions.FACT(double)

CDF Spec: @CalcMgrExcelFACT(number)
```

@CalcMgrExcelFLOOR

Purpose:

Rounds a number down, toward zero

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelMathFunctions.FLOOR(double,double)

CDF Spec: @CalcMgrExcelFLOOR(number, significance)
```

@CalcMgrExcelGCD

Purpose:

Returns the greatest common divisor

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelMathFunctions.GCD(double[])

CDF Spec: @CalcMgrExcelGCD(numbers)
```

@CalcMgrExcelLCM

Purpose:

Returns the least common multiple

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelMathFunctions.LCM(double[])

CDF Spec: @CalcMgrExcelLCM(numbers)
```

@CalcMgrExcelMROUND

Purpose:

Rounds a number to a specified number of digits

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelMathFunctions.MROUND(double,double)

CDF Spec: @CalcMgrExcelMROUND(number, num_digits)
```

@CalcMgrExcelMULTINOMIAL

Purpose:

Returns the multi-nominal of a set of numbers

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelMathFunctions.MULTINOMIAL(double[])

CDF Spec: @CalcMgrExcelMULTINOMIAL(numbers)
```

@CalcMgrExcelODD

Purpose:

Rounds a number up to the nearest odd integer

Syntax:

```
Java Class: com.hyperion.calcmgr.common.excel.cdf.ExcelMathFunctions.ODD(double)

CDF Spec: @CalcMgrExcelODD(number)
```

@CalcMgrExcelPOWER

Purpose:

Returns the result of a number raised to a power

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelMathFunctions.POWER(double,double)

CDF Spec: @CalcMgrExcelPOWER(number, power)
```

@CalcMgrExcelPRODUCT

Purpose:

Multiplies its arguments

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelMathFunctions.PRODUCT(double[])

CDF Spec: @CalcMgrExcelPRODUCT(numbers)
```

@CalcMgrExcelROUNDDOWN

Purpose:

Rounds a number down, towards zero

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelMathFunctions.ROUNDDOWN(double,double)

CDF Spec: @CalcMgrExcelROUNDDOWN(number, num_digits)
```

@CalcMgrExcelROUNDUP

Purpose:

Rounds a number up, away from zero

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelMathFunctions.ROUNDUP(double,double)

CDF Spec: @CalcMgrExcelROUNDUP(number, num_digits)
```

@CalcMgrExcelSQRT

Purpose:

Returns a positive square root

Syntax:

```
Java Class: com.hyperion.calcmgr.common.excel.cdf.ExcelMathFunctions.SQRT(double)

CDF Spec: @CalcMgrExcelSQRT(number)
```

@CalcMgrExcelSQRTPI

Purpose:

Returns the square root of (number * pi)

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelMathFunctions.SQRTPI(double)

CDF Spec: @CalcMgrExcelSQRTPI(number)
```

@CalcMgrExcelSUMPRODUCT

Purpose:

Returns the sum of the products of corresponding array components

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelMathFunctions.SUMPRODUCT(double[],doub
le[])

CDF Spec: @CalcMgrExcelSUMPRODUCT(values1, values2)
```

@CalcMgrExcelSUMSQ

Purpose:

Returns the sum of the squares of the arguments

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelMathFunctions.SUMSQ(double[])

CDF Spec: @CalcMgrExcelSUMSQ(numbers)
```

Statistical Functions

Related Topics

- @CalcMgrExcelAVEDEV
- CalcMgrExcelBINOMDIST
- @CalcMgrExcelDEVSQ
- @CalcMgrExcelLARGE
- @CalcMgrExcelMEDIAN
- @CalcMgrExcelNORMSDIST

- @CalcMgrExcelNORMSINV
- @CalcMgrExcelPERCENTILE
- @CalcMgrExcelPERCENTRANK
- @CalcMgrExcelRANK
- @CalcMgrExcelSMALL
- @CalcMgrExcelSTDEV
- @CalcMgrExcelVAR
- @CalcMgrExcelVARP
- @CalcMgrIsFinite

@CalcMgrExcelAVEDEV

Purpose:

Returns the average of the absolute deviations of data points from their mean

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelStatisticalFunctions.AVEDEV(double[])

CDF Spec: @CalcMgrExcelAVEDEV(numbers)
```

CalcMgrExcelBINOMDIST

Purpose:

Returns the individual term binomial distribution probability

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelStatisticalFunctions.BINOMDIST(double,
double, double, boolean)

CDF Spec:@CalcMgrExcelBINOMDIST(successes, trials, probSuccess, cumulative)
```

Parameters:

`Cumulative`: A logical value that determines the form of the function

`@_true`: A logical value that determines the form of the function. If cumulative is *true*, then `BINOMDIST` returns the cumulative distribution function, which is the probability that there are at most number_s successes

`@_false`: Returns the probability mass function, which is the probability that there are number_s successes

@CalcMgrExcelDEVSQ

Purpose:

Returns the sum of squares of deviations

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelStatisticalFunctions.DEVSQ(double[])

CDF Spec: @CalcMgrExcelDEVSQ(numbers)
```

@CalcMgrExcelLARGE

Purpose:

Returns the nth highest number

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelStatisticalFunctions.NTHLARGEST(doubl
e[],double)

CDF Spec: @CalcMgrExcelLARGE(values, rank)
```

@CalcMgrExcelMEDIAN

Purpose:

Returns the median of the given numbers

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelStatisticalFunctions.MEDIAN(double[])

CDF Spec: @CalcMgrExcelMEDIAN(values)
```

@CalcMgrExcelNORMSDIST

Purpose:

Returns the normal distribution

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelStatisticalFunctions.NORMSDIST(double)

CDF Spec: @CalcMgrExcelNORMSDIST(value)
```

@CalcMgrExcelNORMSINV

Purpose:

Returns the value z such that, with probability p, a standard normal random variable takes on a value that is less than or equal to z. A standard normal random variable has mean 0 and standard deviation 1.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelStatisticalFunctions.NORMSINV(double)

CDF Spec: @CalcMgrExcelNORMSINV(probability)
```

@CalcMgrExcelPERCENTILE

Purpose: Returns the k-th percentile of values in a range

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelStatisticalFunctions.PERCENTILE(doubl
e[],double)

CDF Spec: @CalcMgrExcelPERCENTILE(values,percentile)
```

@CalcMgrExcelPERCENTRANK

Purpose: Returns the rank of a value in a data set as a percentage of the data set

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelStatisticalFunctions.PERCENTRANK(doubl
e[],double)

CDF Spec: @CalcMgrExcelPERCENTRANK(values, percentile)
```

@CalcMgrExcelRANK

Purpose: Returns the rank of a number in a list of numbers

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelStatisticalFunctions.RANK(double,
double[], boolean)

CDF Spec: @CalcMgrExcelRANK(value, values, order)
```

Parameters:

`@_true`: Ranks the value sorted in descending order

`@_false`: Ranks the value sorted in ascending order

@CalcMgrExcelSMALL

Purpose:

Returns the nth smallest number

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelStatisticalFunctions.NTHSMALLEST(doubl
e[],double)

CDF Spec: @CalcMgrExcelSMALL(values, rank)
```

@CalcMgrExcelSTDEV

Purpose:

Estimates standard deviation based on a sample

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelStatisticalFunctions.STDEV(double[])

CDF Spec: @CalcMgrExcelSTDEV(values)
```

@CalcMgrExcelVAR

Purpose:

Estimates variance based on a sample

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelStatisticalFunctions.VAR(double[])

CDF Spec: @CalcMgrExcelVAR(values)
```

@CalcMgrExcelVARP

Purpose:

Estimates variance based on the entire population

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.excel.cdf.ExcelStatisticalFunctions.VARP(double[])

CDF Spec: @CalcMgrExcelVARP(values)
```

@CalcMgrIsFinite

Purpose:

Evaluates the specified member to determine if its value is finite. It returns *false* if the specified number is infinitely large in magnitude (NaN or Infinity); otherwise, it returns *true*.

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.StringFunctions.isFinite(double)

CDF Spec: @CalcMgrIsFinite(doubleNumber)
```

Example:

In the following example, `@CalcMgrIsFinite` evaluates for the members in the FIX statement, to determine if its value is NaN or Infinity. If the value of "5800" is NaN or Infinity, it changes the value to #Missing.

```
FIX ("BaseData", FY13, Plan, Working, "111", @Relative(P_100,0))
    "5800" (
        IF (NOT @CalcMgrIsFinite("5800"))
            "5800" = #Missing;
        ENDIF
```

```
    )
ENDFIX
```

String Functions

Related Topics

- @CalcMgrCompare
- @CalcMgrConcat
- @CalcMgrDecimalFormat
- @CalcMgrDoubleFromString
- @CalcMgrDoubleToString
- @CalcMgrDQuote
- @CalcMgrEndsWith
- @CalcMgrFindFirst
- @CalcMgrFindLast
- @CalcMgrFormatDouble
- @CalcMgrGetListCount
- @CalcMgrGetListItem
- @CalcMgrIndexOf
- @CalcMgrIntegerToString
- @CalcMgrLastIndexOf
- @CalcMgrLowercase
- @CalcMgrMatches
- @CalcMgrMessageFormat
- @CalcMgrPadText
- @CalcMgrUppercase
- @CalcMgrRemoveQuotes
- @CalcMgrRemoveDQuotes
- @CalcMgrRemoveSQuotes
- @CalcMgrReplaceAll
- @CalcMgrReplaceFirst
- @CalcMgrSortAndReturn
- @CalcMgrSortList
- @CalcMgrSortValues
- @CalcMgrSplit
- @CalcMgrSQuote
- @CalcMgrStartsWith
- @CalcMgrStringsToString
- @CalcMgrSubstring

- @CalcMgrTextLength
- @CalcMgrTrim

@CalcMgrCompare

Purpose:

Compares two strings

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.StringFunctions.compare(String,String,boolean)

CDF Spec: @CalcMgrCompare(text1, text2,ignoreCase)
```

@CalcMgrConcat

Purpose:

Concatenates the specified string to the end of this string

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.StringFunctions.concat(String,String)

CDF Spec: @CalcMgrConcat(text1, text2)
```

@CalcMgrDecimalFormat

Purpose:

Returns a formatted string using the specified format string.

For the format specification, see decimal format in Java documentation.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.StringFunctions.decimalFormat(String,double)

CDF Spec: @CalcMgrDecimalFormat(formatString, value)
```

@CalcMgrDoubleFromString

Purpose:

Converts a double from a string

Syntax:

```
Java
Class:com.hyperion.calcmgr.common.cdf.StringFunctions.getDoubleFromString(String)

CDF Spec:@CalcMgrDoubleFromString(text)
```

@CalcMgrDoubleToString

Purpose:

Converts a double to a string

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.MaxLFunctions.doubleToString(double)

CDF Spec: @CalcMgrDoubleToString(doubleNumber)
```

@CalcMgrDQuote

Purpose:

Adds double quotes to text if not in double quotes

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.StringFunctions.dQuote(String)

CDF Spec: @CalcMgrDQuote(text)
```

@CalcMgrEndsWith

Purpose:

Tests if this string ends with the specified suffix

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.StringFunctions.endsWith(String,String)

CDF Spec: @CalcMgrEndsWith(text, suffix)
```

@CalcMgrFindFirst

Purpose:

Find the first substring of this string that matches the given regular expression.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.StringFunctions.findFirst(String,String,boolean)
CDF Spec: @CalcMgrFindFirst(text, regExpr, ignoreCase)
```

@CalcMgrFindLast

Purpose:

Find the last substring of this string that matches the given regular expression.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.StringFunctions.findLast(String,String,boolean)
CDF Spec: @CalcMgrFindLast(text, regExpr, ignoreCase)
```

@CalcMgrFormatDouble

Purpose:

Returns a formatted string using the specified format string.

For the format specification, see print formats in the Java documentation.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.StringFunctions.formatDouble(String,double)

CDF Spec: @CalcMgrFormatDouble(formatString, value)
```

@CalcMgrGetListCount

Purpose:

Returns the number of items in the list

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.StringFunctions.getListCount(String[])

CDF Spec: @CalcMgrGetListCount(list)
```

@CalcMgrGetListItem

Purpose:

Returns the index item from the list

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.StringFunctions.getListItem(String[],int)

CDF Spec: @CalcMgrGetListCount(list,index)
```

@CalcMgrIndexOf

Purpose:

Returns the index within this string of the first occurrence of the specified substring, starting at the specified index

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.StringFunctions.indexOf(String,String,int)

CDF Spec: @CalcMgrIndexOf(text,searchText,begIndex)
```

> **Note**
>
> If you use -1 for the index in either `@CalcMgrIndexOf` or `@CalcMgrLastIndexOf` (below), the entire string is searched.

@CalcMgrIntegerToString

Purpose:

Converts an integer to a string

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.MaxLFunctions.integerToString(int)

CDF Spec: @CalcMgrIntegerToString(integerNumber)
```

@CalcMgrLastIndexOf

Purpose:

Returns the index within this string of the last occurrence of the specified substring, searching backwards and starting at the specified index

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.StringFunctions.lastIndexOf(String,String,int)

CDF Spec: @CalcMgrLastIndexOf(text,searchText,begIndex)
```

@CalcMgrLowercase

Purpose:

Converts text to lower case

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.StringFunctions.toLower(String)

CDF Spec: @CalcMgrLowercase(text)
```

@CalcMgrMatches

Purpose:

Returns true, if the first substring of this string that matches the given regular expression.

For regular expression, see "java.util.regex.Pattern" in the Java docs.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.StringFunctions.matches(String,String,boolean)
CDF Spec: @CalcMgrMatches(text, regExpr, ignoreCase)
```

@CalcMgrMessageFormat

Purpose:

Creates a string with the given pattern and uses it to format the given arguments.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.StringFunctions.messageFormat(String,String[])
CDF Spec: @CalcMgrMessageFormat(text, parameters)
```

@CalcMgrPadText

Purpose:

Fills the text with padding text before or after the text to make up the length.

For example, `@CalcMgrPadText("01",5,"0",@_true)` returns 01000

`@CalcMgrPadText("01",5,"0",@_false)` returns 00001

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.StringFunctions.padText(String,int,String,boolean
)
```

```
CDF Spec: @CalcMgrPadText(text,length,padText,append)
```

@CalcMgrUppercase

Purpose:

Converts text to upper case

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.StringFunctions.toUpper(String)
```

```
CDF Spec: @CalcMgrUppercase(text)
```

@CalcMgrRemoveQuotes

Purpose:

Removes single or double quotes around a text string

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.StringFunctions.removeQuotes(String)
```

```
CDF Spec: @CalcMgrRemoveQuotes(text)
```

@CalcMgrRemoveDQuotes

Purpose:

Removes double quotes around a text string

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.StringFunctions.removeDQuotes(String)
```

```
CDF Spec: CDF Spec: @CalcMgrRemoveDQuotes(text)
```

@CalcMgrRemoveSQuotes

Purpose:

Removes single quotes around a text string

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.StringFunctions.removeSQuotes(String)

CDF Spec: @CalcMgrRemoveSQuotes(text)
```

@CalcMgrReplaceAll

Purpose:

Replaces each substring of this string that matches the given regular expression with the given replacement.

For the regular expression, see the Java documentation for `java.util.regex.Pattern`

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.StringFunctions.replaceAll(String,String,String)

CDF Spec: @CalcMgrReplaceAll(text, regExpr, replacement)
```

@CalcMgrReplaceFirst

Purpose:

Replaces the first substring of this string that matches the given regular expression with the given replacement.

For the regular expression, see the Java documentation for `java.util.regex.Pattern`

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.StringFunctions.replaceFirst(String,String,String
)

CDF Spec: @CalcMgrReplaceFirst(text, regExpr, replacement)
```

@CalcMgrSortAndReturn

Purpose:

Sorts items in the list based on the values and returns the top n elements

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.StringFunctions.sortAndReturn(String[],double[],i
nt,boolean)

CDF Spec: @CalcMgrSortAndReturn(list,values,topN,sortAscending)
```

@CalcMgrSortList

Purpose:

Sorts items in the list

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.StringFunctions.sortList(String[],boolean,boolean
)
```

```
CDF Spec: @CalcMgrSortList(list,caseSensitive,sortAscending)
```

@CalcMgrSortValues

Purpose:

Sorts items in the list based on the values

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.StringFunctions.sortValues(double[],boolean)
```

```
CDF Spec: @CalcMgrSortValues(values,sortAscending)
```

@CalcMgrSplit

Purpose:

Splits the text based on `regex`

For the regular expression, see the Java documentation for `java.util.regex.Pattern`

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.StringFunctions.split(String,String)
```

```
CDF Spec: @CalcMgrSplit(text, regex)
```

@CalcMgrSQuote

Purpose:

Adds single quotes to text if not in single quotes

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.StringFunctions.sQuote(String)
```

```
CDF Spec: @CalcMgrSQuote(text)
```

@CalcMgrStartsWith

Purpose:

Tests if this string starts with the specified prefix

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.StringFunctions.startsWith(String,String)

CDF Spec: @CalcMgrStartsWith(text, prefix)
```

@CalcMgrStringsToString

Purpose:

Converts a string array to a string that uses a delimiter

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.MaxLFunctions.stringsToString(String[],String)

CDF Spec: @CalcMgrStringsToString(strings, seperator)
```

@CalcMgrSubstring

Purpose:

Returns a new string that is a substring of this string.

The substring begins at `startIndex` and extends to the character at `index endIndex` - 1. Thus the length of the substring is `endIndex-startIndex`.

If the `endIndex` is less than zero, then `endIndex` will be the index of the last character.

Syntax:

```
Java Class:
com.hyperion.calcmgr.common.cdf.StringFunctions.substring(String,int,int)

CDF Spec: @CalcMgrSubstring(text, startIndex, endIndex)
```

@CalcMgrTextLength

Purpose:

Returns the length of the text

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.StringFunctions.length(String)

CDF Spec: @CalcMgrTextLength(text)
```

@CalcMgrTrim

Purpose:

Removes leading and trailing white spaces around a text string

Syntax:

```
Java Class: com.hyperion.calcmgr.common.cdf.StringFunctions.trim(String)

CDF Spec: @CalcMgrTrim(text)
```

Working with Smart Lists

Related Topics

- About Smart Lists
  Smart Lists are custom drop-down lists that users access from Planning data form cells in Planning applications.
- Inserting Smart Lists
  Use Smart Lists in Planning business rules, formula components, or script components.

About Smart Lists

Smart Lists are custom drop-down lists that users access from Planning data form cells in Planning applications.

When clicking into data form cells to enter data, you can select items from drop-down lists instead of typing in the cell. You cannot type in cells that contain Smart Lists.

In Calculation Manager, you can insert a Smart List into a variable or into formula, script, condition, and member range components.

Inserting Smart Lists

Use Smart Lists in Planning business rules, formula components, or script components.

Smart Lists are available on Planning data forms, within certain data cells that a Planning administrator specifies. Smart Lists are customized drop-down lists containing options amongst which users can select.

To insert a Smart List:

1. Do one of these tasks:
   - Open the business rule that contains the component into which you want to insert a Smart List. Then select the formula or script component into which you want to insert a Smart List in the business rule's flow chart.
   - Open the formula component into which you want to insert a Smart List.
   - Open the script component into which you want to insert a Smart List.
2. Do one of these tasks:
   - To insert a Smart List into a business rule, on the **Script** tab, click the **Insert a smart list** icon.
   - To insert a Smart List into a formula component, on the **Formula** tab, click in the **Formula** row, select the **Actions** icon, and select **Smart List**.
   - To insert a Smart List into a script component, on the **Script** tab, click the **Insert a smart list** icon.
3. Click **Save**.

Working with Planning Formula Expressions

Use Planning formula expressions in Calculation Manager graphical or script rules.

You can use the following types of formula expressions:

SmartLists

You can include a Smart List as a variable in a formula expression, such as the formula expression, `"Product Channel"=[[Channel.Retail]]`.

`"Product Channel"` is the account with type "Smart List", `Channel` is the Smart List name, and `Retail` is a Smart List entry. If the Smart List ID for Retail is 2, `Channel.Retail` is replaced with a 2 in the member formula (the application treats Smart Lists as numbers). If the Smart List ID for Retail is 2, 2 is put in the calculation, and 2 is stored in the database.

Calculation Manager Syntax:

```
[[SLName.entryname]]
```

Example:

The following syntax:

```
FIX (Mar, Actual, Working, FY15, P_000, "111")
   "Product Channel" =[[Channel.Retail]] ;
ENDFIX
```

returns the following script:

```
FIX (Mar, Actual, Working, FY15, P_000, "111")
   "Product Channel" =2 ;
ENDFIX
```

Dimensions

`Dimension(dimTag)` returns the name of a predefined dimension.

The dimtags are:

- DIM_NAME_PERIOD
- DIM_NAME_YEAR
- DIM_NAME_ACCOUNT
- DIM_NAME_ENTITY
- DIM_NAME_SCENARIO
- DIM_NAME_VERSION

- DIM_NAME_CURRENCY

Calculation Manager Syntax:

```
[[Dimension("DIM_NAME_ENTITY")]]
```

Example:

```
CALC DIM([[Dimension("DIM_NAME_ENTITY")]]);
```

In this application, `Entity` is named *Entity*, so the above script returns:

```
CALC DIM ("Entity");
```

If the entity dimension was named *Cost Center*, it would return:

```
CALC DIM ("Cost Center");
```

Planning User Variables

Planning user variables return the user variable's member.

Calculation Manager Syntax:

```
[[PlanningFunctions.getUserVarValue("xyz")]]
```

Example:

```
FIX (Feb, Actual, Working, P_000, [[PlanningFunctions.getUserVarValue("Entity
View")]] )
   "5800" = 40;
ENDFIX
```

In this application there is a Planning User Variable named *Entity View*. For this user it is set to 112. So the example above the script returns:

```
FIX (Feb, Actual, Working, P_000, "112")
   "5800" = 40;
ENDFIX
```

Periods

Related Topics

- Period(periodName)
- NumberofPeriodsinYear and NumberofYears

Period(periodName)

`Period(periodName)` returns the specified period.

The period name options are:

- FIRST_QTR_PERIOD
- SECOND_QTR_PERIOD

- THIRD_QTR_PERIOD
- FOURTH_QTR_PERIOD
- FIRST_PERIOD
- LAST_PERIOD

Calculation Manager Syntax:

```
[[Period("FIRST_QTR_PERIOD")]]
```

Example:

The following syntax:

```
FIX ( Mar, Actual, Working, P_000, "6100", FY15 )
   "120" =[[Period("FIRST_QTR_PERIOD")]];
ENDFIX
```

returns the following script:

```
FIX (Mar, Actual, Working, P_000, "6100", FY15)
   "120" = "Mar";
ENDFIX
```

NumberofPeriodsinYear and NumberofYears

`NumberofPeriodsInYear` returns the number of periods in the year and `NumberofYears` returns the number of years in the application.

Calculation Manager Syntax:

```
[[NumberOfPeriodsInYear]]
```

```
[[NumberOfYears]]
```

Example:

The following syntax:

```
FIX (Mar, Actual, Working, P_000, "6100", FY15)
   "120"=[[NumberOfPeriodsInYear]];
   "120"=[[NumberOfYears]];
ENDFIX
```

returns the following script:

```
FIX (Mar, Actual, Working, P_000, "6100", FY15)
   "120"=12;
   "120"=9;
ENDFIX
```

Scenarios

Expose Scenario Planning Range time horizon and Module Planning and Forecast Preparation configuration task information as expressions, which allow you to specify the following:

- **Start Year**: Returns the start year in string format for the given scenario.
- **End Year**: Returns the end year in string format for the given scenario.
- **Start Month**: Returns the start month in string format for the given scenario.
- **End Month**: Returns the end month in string format for the given scenario.
- **Module Start Year**: Returns the start year in string format for the given module and scenario.
- **Module End Year**: Returns the end year in string format for the given module and scenario.
- **Module Start Period**: Returns the start period in string for the given module and scenario.
- **Module End Period**: Returns the end period in string format for the given module and scenario.
- **Module Plan Start Year**: Returns an integer value of "1" if *Current Fiscal Year* and "0" if *Next Fiscal Year*.

> **Note**
>
> Module Start Year, Module End Year, Module Start Period, Module End Period, and Module Plan Start Year are only valid for Planning Modules and Strategic Workforce Planning applications and require the "PlanningFunctions" syntax.

Calculation Manager Syntax:

```
[[getStartYear("ScenarioName")]]
[[getEndYear("ScenarioName")]]
[[getStartMonth("ScenarioName")]]
[[getEndMonth("ScenarioName")]]
[[PlanningFunctions.getModuleStartYear("ModuleName","ScenarioName")]]
[[PlanningFunctions.getModuleEndYear("ModuleName","ScenarioName")]]
[[PlanningFunctions.getModuleStartPeriod("ModuleName","ScenarioName")]]
[[PlanningFunctions.getModuleEndPeriod("ModuleName","ScenarioName")]]
[[PlanningFunctions.isPlanStartYearSameAsCurrentFiscalYear("ModuleName")]]
```

Scenario Name

`ScenarioName` can be a typed in Scenario member, or a Calculation Manager run-time prompt member type variable. The member must be enclosed in double quotes. For example, `[[getStartYear("Actual")]]`.

`ScenarioName` can also be a substitution variable in the following formats:

- `getSubVarValue("CubeName", "SubstitutionVariableName")` returns the substitution variable value for a given substitution variable at the cube level defined by the `CubeName`

- `getSubVarValue("SubstitutionVariableName")` returns the substitution variable value for a given substitution variable at the application level for all cubes

The `CubeName` and `SubstitutionVariableName` must be enclosed in double quotes, do not use & or {}. See Example 3 and Example 4 below.

Module Name

`ModuleName` must be a pre-defined name of a current Module in Planning and Strategic Workforce Planning. These expressions using `ModuleName` as a parameter use the Planning and Forecast Preparation configuration tasks time horizon information for `ScenarioName`, which may have different time horizons per Module within an application for the Plan and Forecast Scenarios.

Other Scenarios beyond Plan and Forecast can also be referenced in these expressions, but will only return the time horizon set for the Scenario Planning Range in the dimension editor, which will be the same for all Modules within the application. See About Scenarios for more information on the Scenario Planning Range.

The Plan Start Year is another Planning and Forecast Preparation configuration task that is specific to the Plan Scenario for Modules. This expression returns a value relative to its setting such that it returns an integer value of "1" if set to *Current Fiscal Year* and "0" if set to *Next Fiscal Year*. This expression is only available within the context of a conditional statement as it does not return a dimension member as a value. See Example 5 below.

Valid values for `ModuleName` are as follows and are applicable to the specific business process:

Table 9-4 Valid `ModuleName` Values

Business Process	Valid `ModuleName` Value
Planning Modules	• "Capital" • "Financials" • "Projects" • "Workforce"
Strategic Workforce Planning	• "Strategic Workforce" • "Workforce"

> **Note**
>
> The `ModuleName` value must be enclosed in double quotes and is not case-sensitive.

Example 1

The following syntax, where `{rtpScenario}` is a run-time prompt variable of type *member* with a default value of *actual*:

```
FIX({rtpScenario}, [[getStartYear({rtpScenario})]]:
[[getEndYear({rtpScenario})]],
   [[getStartMonth({rtpScenario})]]:[[getEndMonth({rtpScenario})]])
      FIX ( Working, P_000, "111")
        "5800" = 5500;
      ENDFIX
ENDFIX
```

returns the following script:

```
FIX ("Actual", "FY10" : "FY18", "Jan" : "Dec")
    FIX (Working, P_000, "111")
      "5800" = 5500;
    ENDFIX
ENDFIX
```

Example 2

The following syntax:

```
FIX({rtpScenario}, [[PlanningFunctions.getModuleStartYear("CAPITAL",
{rtpScenario})]]:
[[PlanningFunctions.getModuleEndYear("CAPITAL",{rtpScenario})]] , "Jan" :
"Dec")
  FIX(OEP_Working, P_000, "111")
    "5800" = 5500;
  ENDFIX
ENDFIX
```

returns the following script, where the Capital module Planning and Forecast Preparation configuration task set the Start and End Years for FY18 and FY22, respectively, for the Plan Scenario:

```
FIX("OEP_Plan", "FY18" : "FY22", "Jan" : "Dec")
  FIX(OEP_Working, P_000, "111")
    "5800" = 5500;
  ENDFIX
ENDFIX
```

Example 3

The following syntax uses a substitution variable at the cube level. In this example, `Plan1` is the cube name and `CurrentMonth` is the substitution variable name.

```
FIX ("OEP_Plan", [[getStartMonth(getSubVarValue("Plan1", "CurrentMonth"))]],
"FY15",
"BU Version_1", "No Currency", "No Entity", "No Grades")
"Current" = 15;
ENDFIX
```

This produces the following script:

```
FIX ("OEP_Plan", "Jan", "FY15", "BU Version_1", "No Currency", "No Entity",
"No
Grades")
"Current" = 15;
ENDFIX
```

Example 4

The following syntax uses a substitution variable at the application level. In this example, `CurrentMonth` is the substitution variable name.

```
FIX ("OEP_Plan", [[getStartMonth(getSubVarValue("CurrentMonth"))]], "FY15",
"BU
Version_1", "No Currency", "No Entity", "No Grades")
"Current" = 15;
ENDFIX
```

This produces the following script:

```
FIX ("OEP_Plan", "Jan", "FY15", "BU Version_1", "No Currency", "No Entity",
"No
Grades")
"Current" = 15;
ENDFIX
```

Example 5

The following syntax:

```
FIX("OEP_Plan","OEP_Working",FY20:FY24)
     "Bonus"
     (
     IF([[PlanningFunctions.isPlanStartYearSameAsCurrentFiscalYear
("Workspace")]] == 1)
          "Bonus" = "Salary" * 0.2;
     ELSEIF([[PlanningFunctions.isPlanStartYearSameAsCurrentFiscalYear
("Workforce")]] == 0)
          "Bonus" = "Salary" * 0.3;
     )
ENDFIX
```

produces the following script where the Workforce module Planning and Forecast Preparation configuration task has set the Plan Start Year to *Next Fiscal Year* such that first conditional test (IF) fails and is not executed while the second conditional test (ELSEIF) passes and is executed since the `isPlanStartYearSameAsCurrentFiscalYear` returns an integer of "0" if set to *Next Fiscal Year*:

```
FIX("OEP_Plan","OEP_Working", FY20:FY24)
     "Bonus"
     (
     IF(0 == 1)
          "Bonus" = "Salary" * 0.2;
     ELSEIF(0 == 0)
          "Bonus" = "Salary" * 0.3;
     )
ENDFIX
```

Cross-References

Related Topics

- CrossRef(accountName)
- CrossRef(accountName, prefix)
- CrossRef(accountName, prefix, true)

CrossRef(accountName)

`CrossRef(accountName)` generates a cross reference by adding the default prefix of No to each dimension name (except Currency, Period and Year), followed by the specified account.

Calculation Manager Syntax:

```
[[CrossRef(accountName)]]
```

Example:

Assume the application has the following dimensions: Account, Period, HSP_View, Year, Scenario, Version, Entity, and Product. In this example, the following syntax:

```
FIX (Aug, Actual, Working, FY15, P_000, "112")
   "120" = [[CrossRef("5800")]];
ENDFIX
```

returns the following script:

```
FIX (Aug, Actual, Working, FY15, P_000, "112")
   "120" = "BegBalance"->"No HSP_View"->"No Scenario"->"No Version"->"No
Entity"->"No Product"->"5800";
ENDFIX
```

CrossRef(accountName, prefix)

`CrossRef(accountName, prefix)` generates a cross reference by adding the specified prefix to each dimension name (except Currency, Period and Year), followed by the specified account. The prefix must be in double quotes.

Calculation Manager Syntax:

```
[[CrossRef(accountName, "prefix")]]
```

Example:

Assume the application has the following dimensions: Account, Period, HSP_View, Year, Scenario, Version, Entity, and Product. In this example, the following syntax:

```
FIX (Aug, Actual, Working, FY15, P_000, "112")
   "110" = [[CrossRef("5800", "No")]];
ENDFIX
```

returns the following script:

```
FIX (Aug, Actual, Working, FY15, P_000, "112")
   "110" = "BegBalance"->"No HSP_View"->"No Scenario"->"No Version"->"No
Entity"->"No Product"->"5800";
ENDFIX
```

CrossRef(accountName, prefix, true)

`CrossRef(accountName, prefix, true)` generates a cross reference by adding the specified prefix to each dimension name followed by the specified account. (This includes Year, but not Currency and Period.) The prefix must be in double quotes.

Calculation Manager Syntax:

```
[[CrossRef(accountName, "prefix", true)]]
```

Example:

Assume the application has the following dimensions: Account, Period, HSP_View, Year, Scenario, Version, Entity, and Product. In this example, the following syntax:

```
FIX (Aug, Actual, Working, FY15, P_000, "112")
   "111" = [[CrossRef("5800", "NoX", true)]];
ENDFIX
```

returns the following script:

```
FIX (Aug, Actual, Working, FY15, P_000, "112")
   "111" = "BegBalance"->"NoXHSP_View"->"NoXYear"->"NoXScenario"-
>"NoXVersion"->"NoXEntity"->"NoXProduct"->"5800";
ENDFIX
```

Workforce Cube Year to Date

Related Topics

- CYTD(memberName)
- CYTD(memberName, calTpIndexName, fiscalTPIndexName)

CYTD(memberName)

> **Note**
>
> `CYTD(memberName)` is only for a workforce cube.

`CYTD(memberName)` generates a calendar year-to-date formula for the member

Calculation Manager Syntax:

```
[[CYTD(memberName)]]
```

Example:

```
Fix (NOV, Actual, Working, FY15, P_000, "112")
   "5800" = [[CYTD("6100")]];
ENDFIX
```

CYTD(memberName, calTpIndexName, fiscalTPIndexName)

> **Note**
>
> `CYTD(memberName, calTpIndexName, fiscalTPIndexName)` is only for a workforce cube.

`CYTD(memberName, calTpIndexName, fiscalTPIndexName)` generates a calendar year-to-date formula for the member, and the time period index based on the calendar year and fiscal year. Used when members are renamed. The default member names are "Cal TP-Index" and "Fiscal TPIndex".

Calculation Manager Syntax:

```
[[CYTD(accountName, "Cal TP-Index", "Fiscal TPIndex")]]
```

Example:

```
Fix (Dec, Actual, Working, FY15, P_000, "112")
   "5800" = [[CYTD("6100", "Cal TP-Index", "Fiscal TPIndex")]];
ENDFIX
```

Get ID for String

In Planning, when the account type is *text*, you can write a formula in Calculation Manager to assign a text value.

Calculation Manager Syntax:

```
[[PlanningFunctions.getIdForString("text")]]
```

Example:

In Planning, you have an account named "acct1 text" that is of type *text*. You want to copy your values from FY16 Dec to FY17 Mar, and change the text account to "Not Budgeted."

```
FIX (Actual, Working, P_000, "210")
   DATACOPY FY16->Dec TO FY17->Mar;
   Mar("acct1 text"->FY17 = [[PlanningFunctions.getIdForString("Not
Budgeted")]];)
ENDFIX
```

Working with Hybrid Aggregation in Essbase

Hybrid aggregation for block storage databases means that wherever possible, block storage data calculation executes with efficiency similar to that of aggregate storage databases.

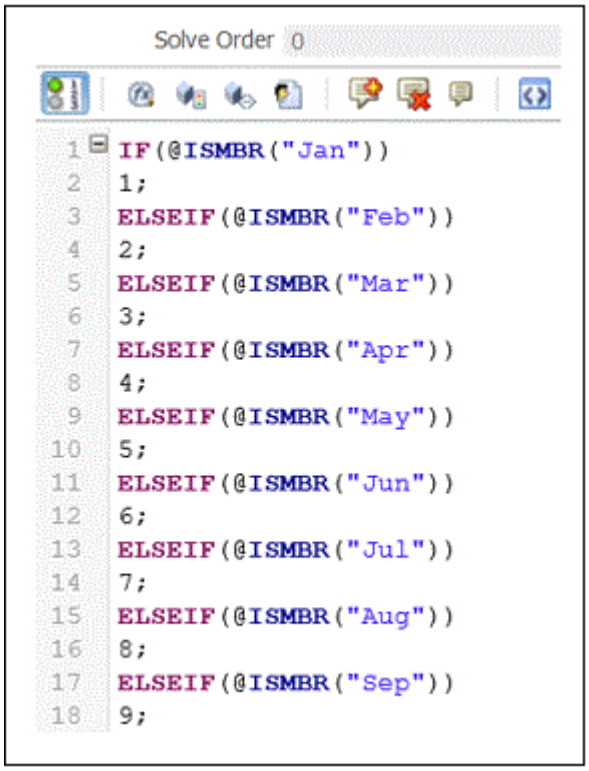
> **Note**
>
> For more information on hybrid aggregation in Oracle Essbase, see the *Technical Reference for Oracle Analytics Cloud - Essbase* guide.

Dynamic Calculations in Hybrid Aggregations

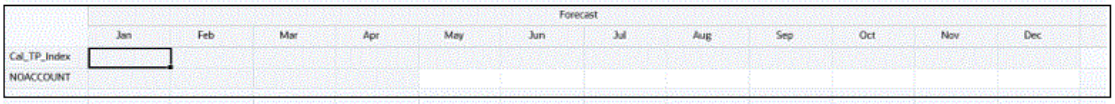
The Oracle Essbase Hybrid Calculator evaluates dynamic member formulas differently from the traditional Essbase calculator.

- The traditional Essbase calculator returns a value for dynamic member formulas for all intersections. Formulas that produce constant values across intersections will only return values when there is a block for that given intersection.
- The Essbase Hybrid Calculator is optimized to ignore constants to improve retrieval performance. This is because the Hybrid Calculator processes dynamic aggregation retrievals from sparse dimensions that are not achievable with traditional BSO.

For example, in the following formula:



An intersection with no data is displayed as follows:



When you add data, the values are displayed as follows:

> **Note**
>
> As long as there is a block, dynamic formulas will evaluate with the Hybrid Calculator.

Calculation Commands Not Support for Hybrid Aggregation

The following calculation commands are not supported for hybrid aggregation mode. If any ofthese commands are encountered in a rule, validation fails and the rule is not launched.

- `CALC ALL`
- `CCONV`
- `CLEARCCTRACK`
- `SET CACHE`
- `SET CCTRACKCALC`

- SET CLEARUPDATESTATUS
- SET DATAIMPORTIGNORETIMESTAMP
- SET LOCKBLOCK
- SET MSG
- SET NOTICE
- SET REMOTECALC
- SET RUNTIMESUBVARS
- SET UPTOLOCAL

Functions Not Supported for Hybrid Aggregation

The following functions are not supported for hybrid aggregation mode. If encountered, Oracle Essbase defaults to block storage execution for these functions.

- @ALLOCATE
- @CREATEBLOCK
- @IRREX
- @MDALLOCATE
- @MDSHIFT
- @MOVSUMX
- @PTD
- @SANCESTVAL
- @STDEV
- @STDEVP
- @@STDEVRANGE
- @SYD
- @TREND
- @XWRITE

10

Validating and Deploying

Related Topics

- Validating Business Rules, Business Rulesets, and Formula and Script Components from the System View
  Validate business rules, rulesets, formula and script components to make sure they are syntactically correct before you deploy them to an application.
- Validating a Business Rule from the Rule Designer
  When you create or debug a business rule, you can validate it in the Rule Designer.
- Deploying Business Rules and Business Rulesets
  Deploy business rules and business rulesets.

Validating Business Rules, Business Rulesets, and Formula and Script Components from the System View

Validate business rules, rulesets, formula and script components to make sure they are syntactically correct before you deploy them to an application.

The validation process ensures that:

- All dimension members are valid for the dimension within the application.
- All functions exist, have the correct number of parameters, and are valid for the application type.
- All variable references in business rules are valid. For replacement variables, the variables are replaced with the correct strings first and then validated. For execution variables, the validation process ensures the variables are defined for the application, the applications within an application type, the plan type, and/or the business rule.
- There are no syntactic errors in the script generation.

If you are validating business rules that have runtime prompts with default values, the validation process ensures that all members in the runtime prompt are valid for the selected plan type and application and that there are no syntactic or semantic errors. If you are validating business rules that have runtime prompts *without* default values, no validation is performed.

> **Note**
>
> If you do not validate rules and rulesets prior to deployment, the deployment may be successful, but the rules and rulesets may fail to launch.

To validate a business rule, a ruleset, or a formula or script component:

1. Do one of these tasks:
   - To validate a ruleset, expand **Rulesets**.

> **Note**
>
> For Planning applications, there is only one Rulesets node for each application at the same level as the plan types and databases.

- To validate a rule, formula, script, or template, expand the calculation type, plan type, or the database, and **Rules**, **Formulas**, **Scripts**, or **Templates**, depending on the object you want to validate.

2. Do one of these tasks:
   - Right-click the object you want to validate, and select **Validate**.
   - Select the object you want to validate, and select **Actions, Validate**.
3. Do one of these tasks:
   a. If the object is validated successfully, click **OK**.
   b. If there are errors, they are displayed. Fix the errors and validate the object again.

Validating a Business Rule from the Rule Designer

When you create or debug a business rule, you can validate it in the Rule Designer.

To validate a business rule from within the Rule Designer:

1. In the System View, expand the application type, the application, the calculation type, plan type, or database, and **Rules**.
2. Right-click the rule you want to validate, and select **Open**.

Deploying Business Rules and Business Rulesets

Deploy business rules and business rulesets.

Related Topics

- About Deploying Business Rules and Business Rulesets
- Making Business Rules and Business Rulesets Deployable and Not Deployable
- Deploying Business Rules and Business Rulesets from the Deployment View
- Deploying a Business Rule or Business Ruleset from the Rule or Ruleset Designer
- Deploying Business Rules with Shortcuts
- Specifying Which Deployed Business Rules Are Displayed in Planning

About Deploying Business Rules and Business Rulesets

You can deploy business rules and business rulesets to Planning. You can deploy one or more business rules or business rulesets (known as a partial deployment), or you can deploy all of the business rules and rulesets in an application (known as a full deployment).

> **Note**
>
> If you have a Planning business rule with a variable that exists at multiple levels (that is, a variable that exists at more than one of these levels: global, application, plan type, or rule), and you delete the variable at the lowest level, you must perform a full redeployment of the Planning application so this deletion is made in any rules that use this variable in Planning. If you perform a partial redeployment only, the deletion of the variable may not be made, and it may still be in use in Planning.

After you deploy business rules and business rulesets to Planning, you can launch them from within data forms or independently from the Launch menu.

See the *Working with Planning* for more information on launching business rules and business rulesets in Planning.

Making Business Rules and Business Rulesets Deployable and Not Deployable

To deploy a subset of business rules and rulesets in an application, you must make them deployable. To make the rules and rulesets deployable, select the check boxes next to their names in **Deployment View.**

> **Note**
>
> To deploy only one business rule or business ruleset, you do not need to make them deployable in **Deployment View**. Instead, you can deploy the rule or ruleset from **System View** by right-clicking it and selecting **Deploy**.

To remove a business rule or ruleset from an application after you deploy it, clear the check box next to its name in **Deployment View**. Then you can perform a full deployment of the application by right-clicking it and selecting **Deploy**.

To make business rules and business rulesets deployable:

1. In **Deployment View**, expand the application type and the application that contains the rule or ruleset to deploy.
2. Expand **To Be Deployed**, and then select the check boxes next to the rules and rulesets to deploy.

   Before a rule or ruleset is deployed, you should validate it for syntactic accuracy. You can validate rules and rulesets manually using the Validate feature. (See Validating Business Rules, Business Rulesets, and Formula and Script Components from the System View.

Deploying Business Rules and Business Rulesets from the Deployment View

You can deploy business rules and business rulesets from the Deployment View. You can also deploy one business rule or one business ruleset to Planning from the Rule Designer (for business rules) or the Ruleset Designer (for business rulesets). See Deploying a Business Rule or Business Ruleset from the Rule or Ruleset Designer.

To deploy business rules and business rulesets from the Deployment View:

1. In **System View**, select **View**, and then **Deployment View**.

> **Note**
>
> You can also deploy business rules and business rulesets from the System View by right-clicking them and selecting Deploy.

2. In the Deployment View, expand the application type.
3. Do one of these tasks:
   - To deploy *all* of the rules and rulesets within an application, select all of the rules or rulesets you want to deploy, right-click on the application, and select Deploy
   - To deploy *a subset* of the business rules and business rulesets (known as partial deployment), expand the application and the **To Be Deployed** node. Then perform these steps:
     a. If they are not selected, select the rulesets you want to deploy.
     b. Expand the plan types that contain rules you want to deploy.
     c. If they are not selected, select the rules you want to deploy.
     d. Right-click, and select **Deploy**.

> **Tip**
>
> To deploy multiple rules or rulesets, use Ctrl + Click and Shift + Click to select them, right-click, and select Deploy.

If the deployment is successful, a "Deployment was successful" message is displayed.

Deploying a Business Rule or Business Ruleset from the Rule or Ruleset Designer

After you design a business rule or ruleset, you can validate and deploy it directly from within the Rule Designer or Ruleset Designer.

To deploy a business rule or business ruleset from the Rule Designer or Ruleset Designer:

1. Do one of these tasks:
   - To deploy a business rule, expand the plan type or the database that contains the rule, then expand **Rules**.
   - To deploy a business ruleset, expand **Rulesets**.
2. Right-click the rule or ruleset you want to deploy, and select **Open**.
3. From the **Rule Designer** or **Ruleset Designer**, select **Actions**. and then **Deploy**.

   If the deployment is successful, a "Deployment was successful" message is displayed.

Deploying Business Rules with Shortcuts

If you have business rules with shortcuts, when you deploy the business rules to applications, a copy of the rule is deployed to each of the applications for which you created a shortcut.

To deploy a business rule with shortcuts:

1. In **System View**, select **View**, and then **Deployment View**.
2. Expand the application type, the application, the **To Be Deployed** node, and the plan type or the database.
3. Right-click the rule you want to deploy, and then select **Deploy All**.

Specifying Which Deployed Business Rules Are Displayed in Planning

Once a business rule has been deployed in Calculation Manager, it can be viewed and run from the Business Rules page in Planning.

You can specify which business rules you want to show up in Planning. To do this:

1. From the **Deployment View**, unselect the rules that you do not want to be displayed in Planning.
2. Right-click on the application node, and select **Deploy**.

   If the rule was previously shown in Planning, but was unchecked in Deployment View in Calculation Manager, after the application is deployed, the rule will no longer show in Planning.

11

Launching Business Rules

Related Topics

- About Launching Business Rules
  You can launch Planning business rules from within the System View or the Rule Designer in Calculation Manager.
- Launching Planning Business Rules and Viewing Logs from the Rule Designer
  You can launch Planning business rules and view the logs that are generated.

About Launching Business Rules

You can launch Planning business rules from within the System View or the Rule Designer in Calculation Manager.

You can also deploy Planning business rules to Planning and then launch them from Planning. For information on launching Planning business rules in Planning, see *Working with Planning*.

> **Note**
>
> - You can enter or edit values for runtime prompt variables when validating, debugging, deploying, analyzing, or launching a rule in Calculation Manager. You can also enter or edit runtime prompt variables when you are validating or deploying business rulesets. If the runtime prompt contains member limits, the validation checks only for valid member names (it does not validate whether the member is within the limit). *Numeric* and *Integer* type variables are validated for runtime prompt limits.
> - Valid combinations and security is not honored when launching a rule in Calculation Manager.
> - For information on desiging runtime prompts to honor Approvals security for members, see About Runtime Prompts and Approvals Security in *Administering Planning*.

Launching Planning Business Rules and Viewing Logs from the Rule Designer

You can launch Planning business rules and view the logs that are generated.

When you have a business rule open for viewing or editing in the Rule Designer, you can launch the rule and view any logs generated in the Log Messages tab of the Rule Designer. You can export the logs to a comma separated values (.csv) file.

To launch business rules and view log messages from the Rule Designer:

1. In **System View**, double-click a rule.
2. In the Rule Designer, select **Actions**, and then **Launch**.

After the rule is run, a confirmation message is displayed that indicates whether the rule launched with or without errors. Click **OK** to close the confirmation message.

3. Select the **Log Message** tab.

   The log messages contain the following information:

   - **Message Number**—ID of the message as shown in the log file
   - **Message Level**—Severity/level of the message
   - **Message Text**—Complete text of the message
   - **Message Timestamp**—Time stamp indicating when the message was generated
   - **Pass Number**—Number of the current pass. The highest/last number is the number of passes in the rule.
   - **Pass Time**—Execution time in seconds. This is the time taken for the current pass.
   - **Cumulative Time**—Total execution time in seconds: This is the total time elapsed since the rule execution began.

     Note: Many log messages are generated from the database at the same time, so the Pass Time (sec) only shows the time for each pass as the time taken for that pass, and the Cumulative Time (sec) shows the time taken from the start of the rule. All other rows show as blank.
   - **Pass %**—Percentage of the total time for that pass of the rule.
   - **Cumulative %**—Percentage of the total time for all passes of the rule. When all the passes are complete, the cumulative percentage should be 100%.

4. **Optional.** Filter the log messages displayed.

   You can filter by the following:

   - **Message Number**—Select a value from the drop down above the **Message Number** column.
   - **Message Level**—Select a value from the drop down above the **Message Level** column.
   - **Message Text**—Enter text in the text box above the **Message Text** column.
   - **Pass Only**—Select **Pass Only** in the drop down above the **Pass #** column to see only the pass numbers and the time each pass took. To return to displaying the compete log information, select the blank option above the **Pass #** column.

> **Note**
>
> After you deploy business rules to Planning, you can also launch them from within Planning. For information, see *Working with Planning*.

> **Note**
>
> You can export the log messages to a comma separated value (.csv) file. See Exporting Log Messages to a File.

12

Exporting and Importing Business Rules, Business Rulesets, Templates, and Formula and Script Components

Related Topics

- About Exporting and Importing
  Export and import objects in your applications.
- Exporting Business Rules, Business Rulesets, Templates, and Formula and Script Components
  When you export an application, an object, or multiple objects, they are exported to an xml file that can be imported into other Calculation Manager applications.
- Exporting Applications
  When you export an application, the application content is saved to an xml file.
- Exporting Log Messages to a File
  When you launch Planning business rules from within Calculation Manager, log messages are generated and displayed in a Log Messages tab within the Rule Designer.
- Importing Rules, Rulesets, Templates, Formulas, and Scripts
  Import rules, rulesets, templates, formulas and scripts into your application with Calculation Manager.

About Exporting and Importing

Export and import objects in your applications.

You can export all of the objects in a Planning application; you can also export individual business rules, business rulesets, templates, and formula and script components within an application.

You can also export business rules, sequences, macros, and variables from Oracle Hyperion Business Rules, and import them into Calculation Manager. Sequences are converted to business rulesets, and macros are converted to templates in Calculation Manager.

After you export applications and objects, you can import them into other Planning applications. For example, you may want to export business rules and business rulesets from an application on a production computer and import them into another application on a test computer.

Exporting Business Rules, Business Rulesets, Templates, and Formula and Script Components

When you export an application, an object, or multiple objects, they are exported to an xml file that can be imported into other Calculation Manager applications.

> **Note**
>
> You can export objects from any view: the System View, the Custom View, and the Deployment View. You can export one object or multiple objects.

To export objects:

1. Do one of these tasks:
   - To export rulesets, expand **Rulesets**.
   - To export rules, formulas, scripts, or templates, expand the plan type, and expand **Rules**, **Formulas**, **Scripts**, or **Templates**
2. Do one of these tasks:
   - To export one object only, right-click it, and then select **Export**.
   - To export multiple objects, select the objects you want to export, right-click, and select **Export**. Use **Shift + Click** and **Ctrl + Click** to select contiguous or non-contiguous objects within different calculation, plan types, or databases, different object types (for example, business rules and formulas), and different applications within an application type.

     After you select Export, you are prompted to open or save the generated .xml file.
3. In **File Download**, do one of these tasks:
   - To view the contents of the generated .xml file, select **Open**.
   - To save the generated .xml file without first viewing it, select **Save**, enter a name for the file (or accept the default), and click **Save** again.

Exporting Applications

When you export an application, the application content is saved to an xml file.

To export an application:

1. Right-click an application, and then select **Export**.
2. In **File Download**, do one of these tasks:
   - To view the contents of the generated xml file, select **Open**.
   - To save the generated xml file without viewing its contents first, select **Save**, enter a name for the file (or accept the default), and click **Save** again.

Exporting Log Messages to a File

When you launch Planning business rules from within Calculation Manager, log messages are generated and displayed in a Log Messages tab within the Rule Designer.

You can export these log messages to a comma separated value (.csv) file. See Launching Planning Business Rules and Viewing Logs from the Rule Designer.

To export log messages generated from launching Planning business rules within Calculation Manager:

1. In **System View**, double-click the rule you want to launch.
2. When the rule opens in the Rule Designer, select **Actions**, and then **Launch**.

   After the rule is run, a confirmation message is displayed that indicates whether the rule launched with or without errors.
3. Click **OK** to exit the confirmation message.

   The log messages are displayed on the Log Messages tab.
4. To export the log messages generated while the rule was launched, select the **Actions**, and then **Export**.

   A file named `RuleLogMessages.csv`, with all of the log messages from the table, is available for download after the export. Save the file and open it using Microsoft Excel with a comma as the separator.

Importing Rules, Rulesets, Templates, Formulas, and Scripts

Import rules, rulesets, templates, formulas and scripts into your application with Calculation Manager.

After you import, you can save the results of the import to a local file.

To import an object, it must be in one of these file types:

- `.xml`, a file that contains objects in xml format
- `.csc`, a file that contains objects in calc script format
- `.zip`, zip files can only contain xml files

To import objects:

1. In **System View**, select **Actions**, and then **Import**, or click  .
2. In the **Import** dialog box, in **File Import Details**, click **Browse**, and then select a file to import.
3. In **Location Details**, enter an application type, application, and cube.
   - If the file is a `.csc` file, you must enter location details.
   - If the file is an `.xml` file, you do not have to enter location details if the location details are already in the import file.

   The information entered in **Location Details** overrides the location specified in the import file. If no location information is specified in the import file, you must enter information in **Location Details**; otherwise, the import fails.

4. In **Import Options**, select one of the following options:
   - **Overwrite Existing Objects**—Imported objects replace the objects in the application and cube.
   - **Skip Existing Objects**—Imported objects are added to the objects in the application and cube as long as the object does not already exist; otherwise, the object is not imported, and the status in the results show "skipped."
   - **Error Out for Duplicates**—If the imported objects duplicate objects that already exist in the application and cube, the names of the duplicate objects are written to the log file, no objects are imported, and the import process stops.
5. Click **Import**.
6. **Optional:** Click **Save As** to save the results of the import to a local file.

13

Administering Essbase Servers, Applications, and Databases for Planning Applications

Related Topics

- Working with Database Properties
  View and edit general, dimension, statistics, transactions, and modifications properties.
- Removing Locks from Database Objects
  You can view and unlock objects, according to your permissions.
- Starting and Stopping Applications
  You can start applications for which you have at least Read permission. Oracle Essbase loads newly started applications into memory on the Essbase Server.
- Starting and Stopping Databases
  When you start databases, Oracle Essbase loads the databases into memory on the Essbase Server.
- Restructuring a Database
  When you restructure a database (for example, by adding a member to a dense dimension), data blocks may need recalculating.
- Verifying an Outline
  You can verify an Oracle Essbase outline to see if it has any errors.
- Clearing Data from the Database
  Clear data from aggregate storage applications, and clear block of data from block storage applications.
- Working With Location Aliases for Block Storage Applications
- Using Query Tracking on Aggregate Storage Databases
  Use query data to select the most appropriate set of aggregate views to materialize for a database.
- Compacting Aggregate Storage Database Outlines
  Compact files to remove the records of deleted members and reduce the outline file size.
- Importing and Exporting Level Zero Data
  Import and export level zero data from ASO and BSO cubes.
- Merging Incremental Data Slices
  You can merge all incremental data slices into the main database slice, or merge all incremental data slices into a single data slice while leaving the main database slice unchanged.
- Aggregating Data
  Calculate aggregations for aggregate storage databases that contain data and to which you are granted Calculation permission.
- Executing the Aggregation Process
  Executing the aggregation process improves retrieval performance.
- Managing Requests
  Use the information in the Sessions window to manage active requests.

- Adding Planning Drill Through Definitions
  In Calculation Manager, you can list, add, edit, and delete these cell drill through definitions for Planning plan types.

Working with Database Properties

View and edit general, dimension, statistics, transactions, and modifications properties.

Related Topics

- Viewing and Editing Database Properties
- General Database Properties
- Dimension Properties
- Statistics Properties
- Transactions Properties
- Modifications Properties

Viewing and Editing Database Properties

You can view and edit the database properties for Planning block storage and aggregate storage applications.

To view or edit database properties:

1. In **System View**, click  (Database Properties).
2. In **Enterprise View**, expand a Planning application and a block storage or an aggregate storage application, and then select a database.

> **Note**
>
> The ASO application names are not the same as the ASO plan types in Planning. Expand the ASO application to see the ASO cube name. To match the ASO cubes, check the cube name in Planning.
>
> For example, the Vision application has two ASO cubes, and they are shown in DB properties as "AVision" with cube "VisASO " and "BVision" with cube "Vis1ASO". The cube names match with ASO plan types in Planning.

3. View or edit the information on the database properties tabs, and then click  .

> **Note**
>
> When you change database properties, you must stop and restart the application. See Starting and Stopping Applications

General Database Properties

General information for the database, including properties in the following areas:

- **General**—Database Type, Database Status (whether it is loaded or not), and Minimum Access Level for the database.
- **Calculation**—*Block Storage applications only*
  - **Aggregate Missing Values**—Aggregates missing values during database calculations.

    By default, during full database calculations, Oracle Essbase does not aggregate missing (#Missing) values. When data is not loaded at parent levels, aggregating missing values may improve calculation performance. For databases for which you have Database Manager permissions, you can choose whether to aggregate missing values.

    If you never load data at parent levels, aggregating missing values may improve calculation performance. If you aggregate missing values and load data at the parent level, the parent-level values are replaced by the results of the database consolidation, even if the results are #Missing values.
  - **Create Blocks on Equation**—Creates a data block for certain member combinations.

    If you create blocks on equation, when you assign a non-constant value to a member combination for which no data block exists, Essbase creates a data block. Creating blocks on equation can produce a very large database.

    When you assign a constant to a member on a sparse dimension, Essbase creates a data block. Therefore, when assigning constants to sparse members (for example, "West = 5"), do not select Create Blocks on Equation.

    When assigning anything other than a constant to a sparse member, if you want blocks created, you must select Create Blocks on Equation. For example, if no data exists for Actuals, a member of the sparse Scenario dimension, you must select Create Blocks on Equation to perform the following allocation: 2002Forecast = Actuals * 1.05;
  - **Two-Pass Calculation**—Recalculates certain members.

    If you select Two Pass Calculation, after a default calculation, members tagged as two-pass are recalculated. The two-pass tag is effective on members of the dimension tagged as accounts and on Dynamic Calc and Dynamic Calc and Store members of any dimension.
- **Data Retrieval Buffers**
  - **Buffer Size**—Size of the retrieval buffer. Used to process and optimize retrievals from the spreadsheet add-in and from report scripts.
  - **Sort Buffer Size**—Size of the retrieval sort buffer
- **Storage**—*Block Storage Applications Only*
  - **Current I/O Access Mode**—Current access mode
  - **Pending I/O Access Mode**—One of these options is configured by default:
    - **Buffered I/O**—Uses the file system buffer cache. If Direct I/O was not specified for the Direction setting in the essbase.cfg file when the database was created, buffered I/O is the default.
    - **Direct I/O**—Bypasses the file system buffer cache and performs asynchronous, overlapped I/Os, providing faster response time and greater potential to optimize cache sizes. If Direct I/O is selected, Essbase attempts to use Direct I/O each time that the database is started. If Direct I/O is not available, Essbase uses Buffered I/O. Select Direct I/O to use cache memory locking or the no-wait (asynchronous) I/O provided by the operating system.
  - **Data Compression**—One of these options is configured by default:

* **Bitmap encoding**—A bitmap is used to represent data cells. Only the bitmap, the block header, and other control information are stored on disk. Bitmap encoding is the most efficient method of compressing data. Essbase stores only non-missing values and does not compress repetitive or zero values. When the database brings a data block into the data cache, it uses the bitmap to recreate missing values and fully expands the block.
* **RLE (Run-Length Encoding)**—Consecutive, repetitive values, including zeros, are compressed, and a record is kept of each repeating value and the number of times that it is repeated consecutively. RLE may be preferable if average block density is not greater than three percent or if the database includes many consecutive zero values or any consecutive, repeating value other than zero.
* **ZLIB**—A data dictionary based on the data being compressed is created. Usually, when data is extremely dense, ZLIB compression provides the best compression ratio. However, under some circumstances, other compression methods may yield better results. With ZLIB compression, the storage space that is saved has little or no relationship to the number of missing cells or the number of contiguous cells of equal value.
* **No Compression**—No compression of data is performed.

Dimension Properties

Dimension information for the database, including:

- Number of dimensions in the database
- *(Block storage databases only)* Type of dimension (dense or sparse)
- Members in dimension
- Members stored
- *(Aggregate storage databases only)* Number of levels in each dimension

> **Note**
>
> Dimension properties are read-only.

Statistics Properties

> **Note**
>
> Statistics properties are read-only.

Statistics for Aggregate Storage Applications

- **General**—General statistical information, including:
  - **Database start time**—Start time according to the time zone of the database server
  - **Database elapsed time**—Elapsed time in hours:minutes:seconds
  - **Number of connections**—Number of connected users

- **Aggregate Storage Statistics**—Storage statics for aggregate databases, including:
  - For each dimension in the application, the number of levels and the bits used to store the levels. (n aggregate storage databases, not all dimension levels are stored.)
  - **Maximum key length (bits)**—Sum of all of the bits used by all of the dimensions. For example, the key for all dimensions contains 20 bits, and the first four bits are used by the Year dimension.
  - **Maximum key length (bytes**)—Number of bytes that the key uses per cell
  - **Number of input-level cells**—Number of cells for level 0 intersections across dimensions, without formulas, into which users can enter data, assuming these are all level zero cells
  - **Number of incremental data slices**—Number of data intersections that can be calculated incrementally (only when necessary) as opposed to immediately
  - **Number of incremental input cells**—Number of input cells that can be calculated incrementally (only when necessary) as opposed to immediately
  - **Number of aggregate views**—Number of views that contain aggregate cells
  - **Number of aggregate cells**—Number of cells that must be calculated when they are requested or retrieved, because they are rolled up from lower level values. Aggregate cell values are calculated for each request, or they can be precalculated and stored on disk.
  - **Number of incremental aggregate cells**—Number of aggregate cells that can be updated only when necessary
  - **Cost of querying incremental data (ratio to total cost)**—Average time for retrieving values from the associated aggregate view
  - **Input-level data size (KB)**—Size, in kilobytes, of the data from all level zero cells
  - **Aggregate data size (KB)**—Size, in kilobytes, of the aggregated data from all aggregated cells
- **Run Time**—Run time statics, including:
  - **Cache hit ratio**—Success rate in locating information in the cache versus having to retrieve it from disk
  - **Current cache size**—Dynamically generated cache size
  - **Current cache size limit (KB)**—Limit, in kilobytes, of the cache size
  - **Page reads since last startup**—Number of index pages that were read since the application was started (either automatically or by the user)
  - **Page writes since last startup**—Number of index pages that were updated since the application was started (either automatically or by the user)
  - **Page size (KB)**—Size of the page, in kilobytes
  - **Disk space allocated for data (KB)**—Total amount of hard disk space, in kilobytes, allocated for data storage
  - **Disk space used by data (KB)**—Total amount of disk space, in kilobytes, used for data storage
  - **Temporary disk space allocated (KB)**—Total amount of temporary disk space allocated for data storage
  - **Temporary disk space used (KB)**—Total amount of temporary disk space used for data storage

> **Note**
>
> *Disk space* is the space used in the `Default` tablespace, and *Temporary disk space* is the space used in the `Temp` tablespace. In both cases, some space within some files may not be used.

Statistics for Block Storage Applications

- **General**—General statistical information:
  - **Database start time**—Start time according to the time zone of the database server
  - **Database elapsed time**—Elapsed time in hours:minutes:seconds
  - **Number of connections**—Number of connected users
- **Blocks**—Statistics about the data block of a block storage database:
  - **Number of existing blocks**—Total number of blocks that exist (contain data)
  - **Block size**—Size, in bytes, of each expanded (decompressed) data block (number of cells * 8; ideally, between 8 and 100 kilobytes). To alter block size, you must change the dense-sparse configuration of the database.
  - **Potential number of blocks**—Maximum number of blocks (derived by multiplying the number of members of one sparse dimension by the number of members of another sparse dimension). For example, the Sample Basic database contains 19 Product members and 25 Market members (not counting shared or label-only members). Because Product and Market are sparse dimensions that store data, there are 19 x 25 = 475 potential data blocks.
  - **Existing level 0 blocks**—Total number of level 0 blocks (blocks whose sparse dimension members have no children) that exist (contain data). Because data can be loaded at upper levels, level 0 blocks and blocks that are created by data input are not necessarily the same.
  - **Existing upper-level blocks**—Total number of non-level 0 blocks that exist (contain data). Upper-level blocks include all combinations of upper-level sparse members plus upper-level combinations that include level 0 sparse members.
  - **Block density (%)**—Average percentage fill of data points within each data block, based on a sample of existing data blocks. Dense-sparse configuration should maximize block density. Maximizing block density, however, may result in proliferation of data blocks. Block size and block proliferation considerations may overshadow the attempt to maximize block density.
  - **Percentage of maximum blocks existing**—Percentage that compares the number of blocks that exist and the number of potential blocks. The percentage is a measure of the sparsity of the database. It is not uncommon for the percentage to be very small; for example, less than one percent.
  - **Compression ratio**—Measure of the compression efficiency of blocks stored on disk. The compression ratio usually indicates block density.
  - **Average clustering ratio**—Fragmentation level of data (.pag) files. The maximum value, 1, indicates no fragmentation. If you are experiencing degraded retrieval, calculation, or data load performance and the clustering ratio value is significantly less than 1, consider forcing a rewrite of data files by exporting and reloading data. Rewriting files defragments the files, resulting in a clustering ratio closer to 1.

  - **Average fragmentation quotient**—Free space in a database. For example, an average fragmentation quotient value of 3.174765 means the database is 3% fragmented with free space. As you update and calculate data, empty spaces occur when a block can no longer fit in its original space and will either append at the end of the file or fit in another empty space that is large enough. The higher the number, the more empty spaces you have; therefore,the longer it takes to get to a particular record. The average fragmentation quotient helps you to decide if a restructure should be performed.

- **Run Time**
  - **Index files**—Total number of index files.
  - **Page files**—Total number of page files.

  An index (.ind) or page (.pag) file reaches a maximum of 2 GB before another one is created. The number of index and page files shows an approximate database size to help troubleshoot performance issues. For example, if you have one index or page file, then the database size is greater than or equal to 2 GB. If you have two index or page files, then the database size is greater than or equal to 4 GB.

Tutorial Video:

Managing BSO Database Properties

Transactions Properties

> **Note**
>
> Transaction properties apply to block storage databases only.

The Transactions tab displays information about the access to the database.

- **Committed access** enables transactions to hold read/write locks on all data blocks involved with a transaction until the transaction completes and commits. The following concurrency options may also be pre-configured:
  - **Wait (seconds)**—Number of seconds that a transaction waits for access to locked data blocks. The default is twenty seconds, but another value, **Indefinitely** or **No wait** may also be pre-configured.
  - **Pre-image access**—Users have read-only access to data blocks that are locked for the duration of another concurrent transaction.
- **Uncommitted access** enables transactions to hold read/write locks on a block-by-block basis (the default setting). The synchronization point may also be pre-configured in the following areas:
  - **Commit blocks**—Number of data blocks updated before Oracle Essbase performs a commit
  - **Commit rows**—Number of rows of a data file processed during a data load before Essbase performs a commit

Modifications Properties

The Modifications tab displays information about the most recent operation (outline update, data load, or calculation) performed against the database:

- **Operation**—Type of operation, such as data load or calculation
- **User**—Name of the user who performed the operation
- **Start Time**—Time, according to Essbase Server, that the operation began, including preparatory tasks, such as locking data (For the duration of the operation, see the elapsed time entry in the application log.)
- **End Time**—Time, according to Essbase Server, that the operation ended
- **Note**—Optional comment

Removing Locks from Database Objects

You can view and unlock objects, according to your permissions.

Users with Administrator permissions can unlock any object. Users without Administrator permissions can unlock only objects that they locked.

The server uses a check out facility for database objects (such as calculation scripts, report scripts, and business rules files) to ensure that objects are modified by only one user at one time.

By default, objects are locked when you open them to modify them, and then unlocked when you close the object.

Objects in Calculation Manager may be locked when you perform actions on them. To unlock objects, you select the plan type for which you want to unlock objects in the Enterprise View.

To remove a lock from a database:

1. In **System View**, click the **Database Properties** icon.
2. In **Enterprise View**, expand the Planning application type and the application that contains the database for which you want to remove a lock.
3. Right-click the database, and then select **Remove Lock(s)**.

Starting and Stopping Applications

You can start applications for which you have at least Read permission. Oracle Essbase loads newly started applications into memory on the Essbase Server.

You can specify that databases start when their parent applications start. In this case, if you start an application before users connect to the databases within the application, users may experience better initial performance (upon database connection) because the application and all associated databases are in memory.

When you stop applications, Essbase unloads the applications and all databases within the applications from memory on the Essbase Server. Thus, available memory is increased. To ensure that databases within applications are not corrupted, you must stop applications properly.

To start or stop an application:

1. In **System View**, click the **Database Properties** icon.
2. In **Enterprise View**, expand the Planning application type and the application you want to start or stop
3. Right-click the application, and select:
   - **Start Application**
   - **Stop Application**

> **Note**
>
> You need to stop and then restart applications any time you make changes to database settings.

Starting and Stopping Databases

When you start databases, Oracle Essbase loads the databases into memory on the Essbase Server.

Index caches are allocated automatically, and data-file and data caches are allocated when blocks are requested. If you start databases before users access them, users may experience better initial performance (upon connection) because the databases are in memory.

When you start databases from applications that are not started, the applications and all databases within the applications are loaded. You can start one database or all databases for an application.

When you stop databases, Essbase unloads the databases from memory on the Essbase Server and commits updated data to disk. Thus, on the server computer, you increase available memory.

You can stop one database or all databases for an application.

> **Note**
>
> You can start and stop block storage application databases.

To start or stop a database:

1. In **System View**, click the **Database Properties** icon.
2. In **Enterprise View**, expand the Planning application type and the application that contains the database you want to start or stop
3. Right-click the database, and select:
   - **Start Database**
   - **Stop Database**

To start or stop all databases for an application:

1. In **System View**, click the **Database Properties** icon.
2. In **Enterprise View**, expand the Planning application type and the application that contains the databases you want to start or stop

3. Right-click the application, and select:
   - **Start All Databases**
   - **Stop All Databases**

Restructuring a Database

When you restructure a database (for example, by adding a member to a dense dimension), data blocks may need recalculating.

Oracle Essbase marks all data blocks as dirty. When you calculate the restructured database, all blocks are calculated.

> **Note**
>
> You can restructure block storage application databases.

In the following cases, you should restructure the database:

- Add, delete, or move a dense dimension Dynamic Calc and Store member.
- Change a dense dimension Dynamic Calc and Store member to a Dynamic Calc member.
- Change a dense dimension Dynamic Calc member to a Dynamic Calc and Store member.
- Change the storage property of a nondynamic member in a dense dimension to Dynamic Calc.
- Change the storage property of a dense dimension from Dynamic Calc member to a nondynamic value.
- Change the storage property of a nondynamic member in a sparse dimension Dynamic Calc or Dynamic Calc and Store.

To restructure a database:

1. From the **System View**, click the **Database Properties** icon.
2. In the **Enterprise View**, expand the Planning application type, the application, and the plan type whose database you want to restructure.
3. Right-click the plan type, and select **Restructure Database**.
4. Confirm whether you want to restructure the database.

   Calculation Manager displays a message letting you know whether the database was restructured successfully.

Verifying an Outline

You can verify an Oracle Essbase outline to see if it has any errors.

The outline verification process takes into account the outline type (aggregate storage or block storage) and verifies the outline according to the rules for each type. Once an outline is error free, member formulas are verified.

To verify an outline:

1. From the **System View**, click the **Database Properties** icon.

2. In the **Enterprise View**, expand the Planning application type, the application, and the plan type whose outline you want to verify.
3. Right-click the plan type, and select **Verify Outline**.

When verifying an outline, Essbase checks the following items:

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

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

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

Clearing Data from the Database

Clear data from aggregate storage applications, and clear block of data from block storage applications.

Related Topics

- Clearing Data From Aggregate Storage Applications

- Clearing Blocks of Data From Block Storage Applications

Clearing Data From Aggregate Storage Applications

To clear data from an aggregate storage application:

1. In **System View**, click  .
2. In **Enterprise View** in the **Database Properties** tab, expand an aggregate storage application.
3. Right-click an ASO cube, then select **Clear**, and then select an option:
   - **All Data**—Clears all data from the database
   - **All Aggregations**—Clears all aggregated data values from the database
   - **Partial Data**—Clears data from a specified region in an aggregate storage database, and retains the data located in other regions

     When you select to clear partial data, enter an MDX expression to define the region to clear, and select either **Logical** or **Physical** to specify type of data clear

     – **Logical**—Data in the specified region is written to a new data slice with negative, compensating values that result in a value of zero for the cleared cells.

     – **Physical**—Data in the specified region is physically removed from the aggregate storage database.

     To use the Member Selector to create MDX syntax, click  . In the **Member Selector** dialog box, select a member or members, or use a function from each dimension, and then click **OK** to create the MDX expression.

     For example:

     ```
     Crossjoin(Crossjoin(Crossjoin(Crossjoin(Crossjoin(Crossjoin(Crossjoin({[NI
     ].Levels(0).Members},{except(DESCENDANTS([Q1]),{[Q1]})}),{[FY12],[FY13],
     [FY14]}),{[Plan]}),{[Working]}),{[100].Levels(0).Members}),
     {[P_TP].Levels(0).Members}),{[FY06]}
     ```

     Note that the MDX expression is validated when you click **OK** to clear the data. If you modify the MDX expression directly, it will validate before running.

> **Note**
>
> You can only clear data from databases for which you have permissions.

Clearing Blocks of Data From Block Storage Applications

To clear blocks of data from a block storage application:

1. In **System View**, click  .
2. In **Enterprise View** in the **Database Properties** tab, expand a block storage application.
3. Right-click a BSO cube, then select **Clear**, and then select an option:
   - **All Data**—Clears all data from the database.

- **Upper level blocks**—Clears only upper-level data blocks. The data values for upper-level blocks are set to #Missing. Upper-level blocks are created for sparse member combinations of which at least one sparse member is a parent member.
- **Non-input blocks**—Clears only data blocks that contain values that are derived from calculations (non-input blocks). When you clear non-input blocks, data values for non-input (calculated) cells are set to #Missing.
- **Dynamic blocks**—Clears only data blocks that are dynamically calculated and stored (Dynamic Calc and Store members).
- **Empty blocks**—Clears only empty blocks of data (blocks where all of the values are #Missing).

> **Note**
>
> You can only clear data from databases for which you have permissions.

Working With Location Aliases for Block Storage Applications

Related Topics

- About Location Aliases
- Displaying a List of Location Aliases
- Exporting a Location Alias

About Location Aliases

A location alias is a descriptor for a data source. A location alias maps an alias name for a database to the physical location of that database. A location alias is set at the database or the application level and specifies an alias, a server, an application, a database, a user name, and a password. You can display and export location aliases.

> **Note**
>
> Location aliases do *not* apply to aggregate storage databases.

Displaying a List of Location Aliases

To display a list of location aliases for a block storage application:

1. In **System View**, click  (Database Properties).
2. In **Enterprise View**, right-click a block storage application, and then select **Location Aliases**.
3. Click **OK** to confirm that you want to start the database, and then click **OK** in the **Database Start Action Status** dialog box.

Exporting a Location Alias

To export a location alias:

1. In **System View**, click  (Database Properties).
2. In **Enterprise View**, right-click a block storage application, and then select **Location Alias**.
3. Click **OK** to confirm that you want to start the database, and then click **OK** in the **Database Start Action Status** dialog box.
4. In the **Location Alias** dialog box, click  (Export), then select the file to export, and then click **OK**.

Using Query Tracking on Aggregate Storage Databases

Use query data to select the most appropriate set of aggregate views to materialize for a database.

You enable query tracking to capture data about the cost of each query that is performed against the database. The cost of a query is an estimate of the average retrieval time required to retrieve values from the view. For the first view (selected by default), the estimation is the average of all possible queries. For views for which query tracking is used, the estimation is the average of the tracked queries. Therefore, a view may, under different conditions, display different estimates. To compute a percentage that evaluates the benefit of using a particular view, divide the query cost value for the view into the query cost value for using views that contain only level 0 values.

Once enabled, query tracking continues until one of the following happens:

- Query tracking is disabled for the database, as described in this topic.
- The application is shut down. If the application is shut down, query tracking does not resume automatically when the application is restarted.
- Additional aggregate views are materialized for the database. Because query tracking data becomes invalid when additional views are materialized, materializing any new aggregate views resets the query tracking.

Query tracking, which is stored only in memory, includes queries from Oracle Hyperion Web Analysis, the grid API, report scripts, Java APIs, and so forth.

> **Note**
>
> Query tracking can be used only on aggregate storage databases.

To enable or disable query tracking:

1. In **System View**, click 
2. In **Enterprise View**, right-click the plan type, then select **Query Tracking**, and then select and one of these options:
   - **Enable**, to enable query tracking
   - **Disable**, to disable query tracking

When you enable query tracking, the database records query information; when you disable query tracking, the database stops recording query information and clears query data from memory.

3. Click **OK** to confirm that you want to enable or disable query tracking.

Compacting Aggregate Storage Database Outlines

Compact files to remove the records of deleted members and reduce the outline file size.

For example, as aggregate storage outline (`.otl`) files are changed when members are added or deleted, the files may increase in size. After the outline file is compacted, the file continues to grow as before.

Compacting the outline file causes the database to restructure the outline. Compacting the outline does not cause the database to clear the data.

When a member is deleted from the outline, the corresponding record of that member in the outline file is marked as deleted, but the record remains in the outline file. Compacting the outline file does not remove the records of deleted members.

> **Note**
>
> You can only compact aggregate storage database outlines. The process of compacting an outline can take place only when no other users or processes are actively using the database

To compact an aggregate storage database outline:

1. In **System View**, click 
2. In **Enterprise View**, right-click the plan type that contains the database whose outline you want to compact, and then select **Compact Outline**.
3. Click **OK** to confirm you want to compact the database outline.

   The **Compact Outline Action Status** dialog displays the progress of the compaction. When the outline is compacted, the **Compact Outline Action Status** dialog displays a success message.
4. In the **Compact Outline Action Status** dialog, click **Show Details** to see details of the outline compaction or **OK** to close the dialog.

Importing and Exporting Level Zero Data

Import and export level zero data from ASO and BSO cubes.

Related Topics

- Importing Level Zero Data from an ASO Cube
- Exporting Level Zero Data from an ASO Cube
- Importing Level Zero Data from a BSO Cube
- Exporting Level Zero Data from a BSO Cube

Importing Level Zero Data from an ASO Cube

To import level zero data from an ASO cube:

1. In **System View**, click  .
2. In **Enterprise View**, in the **Database Properties** tab, expand an aggregate storage application.
3. Right-click an ASO cube, and then select **Import Level Zero Data**.
4. In the **Import Level Zero Data in Cube** dialog box, enter the following information, and then click **OK**:
   - **Zip File Name**—Zip file name for the imported data.
   - **Duplicate Aggregation Method**—Define how to combine multiple values for the same cell.
     - **Add duplicate values**—Add values when the buffer contains multiple values for the same cell.
     - **Assume values equal**—Verify that multiple values for the same cells are identical; if they are, ignore the duplicate values. If the values for the same cell differ, stop the data load with an error message.
     - **Use last value**—Combines duplicate cells by using the value of the cell that was loaded last into the load buffer. This option is intended for relatively small data loads of up to 10,000s of cells. When using this option, data loads are significantly slower, even if there are not any duplicate values.
   - **Options**—You can select the following options if desired:
     - **Ignore missing values**—Ignore #MISSING values in the incoming data stream.
     - **Ignore zero values**—Ignore zeros in the incoming data stream.
5. In the **Level Zero Data Import Status** dialog box, click **Show Details** to show the details of the import, and then click **OK**.

   Once you have imported the data, you can use Planning forms to see the data, or you can look at the cube statistics to see how the values changed.

> **Note**
>
> If the imported data contains Planning Textual values, Smart list values, or currency values from a source other than where the data is being imported, the data may be corrupted.

Exporting Level Zero Data from an ASO Cube

To export level zero data from an ASO cube:

1. In **System View**, click  .
2. In **Enterprise View**, in the **Database Properties** tab, expand an aggregate storage application.
3. Right-click an ASO cube, and then select **Export Level Zero Data**.

4. In the **Export Level Zero Data of Cube** dialog box, enter a zip file name for the exported data, and then click **OK**.

To see the .zip file containing the exported level zero data in Planning:

1. On the Planning Home page, click **Application**, and then click **Overview**.
2. On the **Application** page, click **Actions**, and then click **Inbox/Outbox Explorer**.
3. In the **Inbox/Outbox Explorer**, click the Actions icon next to the .zip file, and then select **Download File**.

Importing Level Zero Data from a BSO Cube

To import level zero data from an BSO cube:

1. In **System View**, click  .
2. In **Enterprise View**, in the **Database Properties** tab, expand a block storage application.
3. Right-click a BSO cube, and then select **Import Level Zero Data**.
4. In the **Import Level Zero Data in Cube** dialog box, enter the zip file name for the imported data, and then click **OK**.

Exporting Level Zero Data from a BSO Cube

To export level zero data from a BSO cube:

1. In **System View**, click  .
2. In **Enterprise View**, in the **Database Properties** tab, expand an aggregate storage application.
3. Right-click a BSO cube, and then select **Export Level Zero Data**.
4. In the **Export Level Zero Data of Cube** dialog box, enter a zip file name for the exported data, and then click **OK**.

To see the .zip file containing the exported level zero data in Planning:

1. On the Planning Home page, click **Application**, and then click **Overview**.
2. On the **Application** page, click **Actions**, and then click **Inbox/Outbox Explorer**.
3. In the **Inbox/Outbox Explorer**, click the Actions icon next to the .zip file, and then select **Download File**.

Merging Incremental Data Slices

You can merge all incremental data slices into the main database slice, or merge all incremental data slices into a single data slice while leaving the main database slice unchanged.

To merge slices, you must have the same privileges as for loading data (Administrator or Database Manager permissions).

> **Note**
>
> You can only merge incremental data slices for aggregate storage databases only.

To merge incremental data slices:

1. In **System View**, click 
2. In **Enterprise View**, right-click the plan type that contains the database whose data you want to merge, then select **Merge Data**, and then select one of these options:
   - **All**, to merge all data slices into one
     - Keep cells with zero value (default)
     - Remove cells with zero value
   - **Incremental**, to merge incremental data slices into one and do either of these:
     - Keep cells with zero value (default)
     - Remove cells with zero value
3. In the **Confirm Merge Data Action** dialog, click **OK** to confirm you want to merge the data.

Aggregating Data

Calculate aggregations for aggregate storage databases that contain data and to which you are granted Calculation permission.

To perform an aggregation, you use system recommended views. The selection of views and aggregation processes are combined into one, non-configurable operation performed by the server. You can optionally specify the maximum disk space for the resulting files, base the view selection on user querying patterns, and include rollup hierarchies in the view selection.

> **Note**
>
> You can aggregate data for aggregate storage databases only. For an example of best practices when aggregating data, see Executing the Aggregation Process.

To perform an aggregation:

1. In **System View**, click 
2. In **Enterprise View**, right-click the plan type that contains the database whose outline you want to compact, and then select **Execute Aggregation**.
3. In the **Execute Aggregation Action - Use Recommended Views** dialog box, select one of the following options:
   - **Based on query data?**—Aggregate the views the server selects, based on collected user querying patterns. This option is only available if query tracking is turned on.
   - **Include rollup option?**—Include secondary hierarchies (with default level usage) in the view selection process.

- **Include growth size option?**—Aggregate the views the server selects, until the maximum growth of the aggregated database exceeds limits you specify. Enter the size (in megabytes) beyond which the server should stop the aggregation.

4. Click **OK**.

   If existing aggregation data exists, a message is displayed asking if you want to delete existing aggregations and rerun the aggregation process. If existing data exists, it is deleted before the aggregation process is rerun.

5. Click **OK** to delete existing aggregation data and rerun the aggregation.

Executing the Aggregation Process

Executing the aggregation process improves retrieval performance.

ASO cubes do not use calculation scripts to aggregate the data; instead, ASO attempts to dynamically calculate upper level members. This can result in faster batch processing time, but might result in longer retrieval times. You can improve this situation by turning on Query Tracking to capture queries against the ASO cube for operations such as working with forms and running ad-hoc reports. These queries are used in aggregate process, which tells Oracle Essbase to use the query patterns picked up by Query Tracking to build the Aggregation views. Once the aggregate views are created, you should see an improvement in retrieval performance.

Before executing the aggregation process, do the following:

- Merge Incremental Data Slices and Remove Zero Value Cells
- Enable Query Tracking
- Perform Actions to Create Queries
- Execute Aggregation Using Query Tracking

Merge Incremental Data Slices and Remove Zero Value Cells

Some Oracle Essbase operations like logical clear and load data may create incremental data slices with zero value cells. Essbase does not allow aggregation with incremental data slices. As a result, you may need to perform a merge operation to merge incremental data slices.

To perform a merge operation and remove zero value cells:

1. In **System View**, click  , and then select an Aggregate Storage database to load its properties.
2. In **Enterprise View**, right-click the cube, then select **Merge Data**, then **All**, and then **Remove Cells With Zero Value**.
3. Click **OK** to confirm the merge data action.

   This merges all incremental slices into the main database slice and removes cells that have a value of zero. (Logically clearing data from a region, results in a cell with a value of zero). As a result, the database size is significantly reduced.

   If a merge is not needed, a message is displayed stating that "There is no incremental data or it is already merged. The specified merge is not necessary." (Click **Show Details** in the **Merge Action Status** dialog box to see the full message.)

Enable Query Tracking

You can enable query tracking for ASO databases to record a meaningful set of queries, and then use the recorded query data to select the most appropriate set of aggregate views to materialize for that database.

The cube refresh operation done in Planning performs an outline restructure operation. As part of the restructure operation, Oracle Essbase removes the tracked queries. A merge operation also removes the tracked queries.

To enable query tracking:

1. In **Enterprise View**, expand **Planning**, then expand the ASO application, then right-click the ASO cube, and then select **Set Query Tracking**.
2. Click **OK** to confirm the query tracking action.
3. Click **OK** in the **Information** dialog box that informs you that query tracking on the database was enabled successfully.

Once you enable Query Tracking, it can be disabled by repeating the steps above. If you disable query tracking, the **Information** dialog box informs you that Query Tracking in enabled, and asks you if you want to disable query tracking, stop the application, or execute the Aggregation process.

Perform Actions to Create Queries

Query Tracking, which is stored only in memory, tracks queries for operations such as opening forms using the ASO cube in Planning, and executing ad-hoc reports.

After you enable query tracking, you can continue to load forms, execute business rules, and run reports. Oracle Essbase will continue to track the queries and store the statistics. You can use these tracked statistics while doing aggregation.

Execute Aggregation Using Query Tracking

Once you have run a sufficient amount of queries, you can execute the aggregation process using the query tracking information.

To execute the aggregation process using query tracking information:

1. In **Enterprise View**, expand **Planning**, then expand the ASO application, then right-click the ASO cube, and then select **Execute Aggregation**.
2. In the **Execute Aggregation Action** dialog box, select **Based on Query Data**, and then click **OK**.

   This operation might take some time to complete.

Calculation Manager checks for the following conditions that can potentially cause the execute aggregation process to fail:

- If query tracking is not enabled or there is no query tracking data, a warning message is displayed. You will not be able to the use query tracking option, but you can still process the aggregation. To use query tracking information, you must enable the query tracking or perform operations that generate query tracking information.
- Oracle Essbase does not allow Aggregate views to be created on an ASO cube when multiple database slices exist. A warning message is displayed, and you will not be able to proceed with aggregation until slices are merged.

In this case, in **Enterprise View**, right-click the cube node and select **Merge Data**, then **All**, and then **Remove Cells With Zero Value**. Merge clears all the tracked query information. As a result, you must perform operations to generate query tracking information.

- If aggregations already exist, a warning message is displayed.
  In this case, click **OK**, and then in the Info dialog box, do one of the following:
  - Click **OK** to drop existing aggregations and rerun the aggregation process.
  - Click **Cancel**, then clear the existing aggregations, and then perform operations to generate query tracking information.

  To clear aggregations:

  1. Right-click an ASO cube, then select **Clear**, and then **All Aggregations**.
  2. In the **Confirm Database Aggregates Clear** dialog box, click **OK**.

Managing Requests

Use the information in the Sessions window to manage active requests.

The Sessions window lists active user sessions and requests for the server, application, or database. A user can have more than one session open at any given time. For example, one user may have open sessions on two databases.

If you have Administrator or Application Manager permissions, you can end all requests, end all requests for a user, or end a specific request.

To end requests:

1. In **System View**, click 
2. In **Enterprise View**, right-click an application, and then select **Sessions**.

   The **Sessions** window displays a list of active sessions and requests. If you have Administrator permissions, the window lists active user sessions for all users on the server. If you have Application Manager permissions, the window lists active sessions for all users, including yourself, who are connected to any application for which you have Application Manager permissions.
3. To log off a user or users, under **Options**, from the **Action** dropdown, select **Log Off**, and then do one of the following tasks:
   - From **Entity**, select **selected user**, and select the user you want to log off. Then click **Apply** to log the user off.
   - From **Entity**, select **all users**, and then do one of the following tasks:
     - From **Source**, select **on selected server** to log off all users on the selected server. Then click **Apply**.
     - From **Source**, select **on selected application** to log off all users on the selected application. Then click **Apply**.
     - From **Source**, select **on selected database** to log off all users on the selected database. Then click **Apply**.
   - From **Entity**, select **all instances of user**, and then do one of the following tasks:
     - From **Source**, select **on selected server**, and select the user instances you want to log off. Click **Apply** to log off all instances of the user from the server.

  - From **Source**, select **on selected application**, and select the user instances you want to log off. Click **Apply** to log off all instances of the user from the application.
  - From **Source**, select **on selected database** , and select the user instances you want to log off. Click **Apply** to log off all instances of the user from the database.

4. To end a request or requests, under **Options**, from the **Action** dropdown, select **Kill**, and then do one of the following tasks:
   - From **Entity**, select **selected request**, and select the request you want to end from the list of sessions. Then click **Apply** to end the selected request.
   - From **Entity**, select **all requests**, and then do one of the following tasks:
     - From **Source**, select **on selected server** to end all requests from all users on the selected server. Then click **Apply**.
     - From **Source**, select **on selected application** to end all requests from all users on the selected application. Then click **Apply**.
     - From **Source**, select **on selected database** to end all requests from all users on the selected database. Then click **Apply**.
   - From **Entity**, select **all requests from user**, and then do one of the following tasks:
     - From **Source**, select **on selected server**, and select a user's request from the list of sessions. Click **Apply** to end all requests from this user on the server.
     - From **Source**, select **on selected application**, and select a user's request from the list of sessions. Click **Apply** to end all requests from this user on the application.
     - From **Source**, select **on selected database** , and select a user's request from the list of sessions. Click **Apply** to end all requests from this user on the database.
5. To display and hide columns, select **View** and then **Columns**. Do any of the following:
   - Select **Show All** to display all columns. By default, all columns are displayed except for the **Connection Source** column.
   - Select **Manage Columns** to move columns between **Hidden Columns** and **Visible Columns** lists.
6. To reorder the columns, select **View** and then **Reorder Columns**. In the **Reorder Columns** dialog, use the up and down arrow keys to reorder the columns as you would like.
7. To sort the list of sessions by column:
   - To sort a column in ascending order, click the column header, or click the **Up** arrow beside the column header.

     For example, to sort the **User** column in alphabetical order, click the column header.
   - To sort a column in descending order, press **Shift** and click the column header, or click the **Down** arrow beside the column header.

     For example, to sort the **Login Time** column so that the longest login time appears first in the list, press **Shift** and click the column header.
8. To refresh the list of sessions, click **Refresh**. For example, if you end a session that shows in process, and then click **Refresh**, the session list shows that the session is no longer in process.
9. To export sessions data to a Microsoft Excel spreadsheet, click the **Export** button, either open or save the file, and click **OK**.

Adding Planning Drill Through Definitions

In Calculation Manager, you can list, add, edit, and delete these cell drill through definitions for Planning plan types.

If you are working with a Planning plan type, and a Planning form that contains members whose data is loaded from a source such as Data Management, you can drill through to view more details for the cell data source.

To add a Planning drill through definition:

1. In **System View**, click  (Database Properties).
2. In **Enterprise View**, right-click an application, and then select **Drill Through Definitions**.

   If asked to confirm the whether you want to start the database, click **OK,**
3. Click **OK** to confirm that you want to start the database, and then click **OK** in the **Database Start Action Status** dialog box.
4. In the **Drill Through Definitions** dialog box, click  (Add).
5. In **Create Drill Through Definition**, create the drill through definition by entering the following information:
   - **URL Name**—Name to identify the drill through definition
   - **XML Contents**—XML to define the URL link

     Enter the URL without the server and port information. The URL must contain the parameter name and column name from the `TDATASEG` table enclosed in the symbol `$`. For example, enter: `LEDGER_ID=$ATTR1$&GL_PERIOD=$ATTR2$`. In this example the value of `ATTR1` is passed as a value for the `LEDGER_ID` parameter, and `ATTR2` is passed as the value for the `GL_PERIOD` parameter. Parameters are separated by the `&` character.

     To specify the request-response between a client and server for the drill URL format, enter either:

     – `GET`—Encodes form data into the URL

       For example, enter: `GET@http://www.oracle.com/`. If no method is entered, then `GET` is the assumed request response.

     – `POST`—Displays form data in the message body

       For example, enter: `POST@http://www.oracle.com/`

     As you enter XML contents, you can click  to import a file into the XML Contents area, and you can click  to export the XML content to Microsoft Excel.
   - **Level 0 Flag**—Whether the URL applies only to level 0 descendents of the region.

     For example, if the Level 0 Flag is enabled for the drillable region `DESCENDANTS("Market"),@CHILDREN(Qtr1)`, then the URL is applicable for all states of "Market" during all months of "Qtr1," and for all level 0 members across the remaining dimensions.
   - **Regions**—Member specifications defining areas of the database that should allow drill-through using the specified URL

You define drillable regions using a member specification of members from one or more dimensions. Define the member specification using the same Oracle Essbase member-set calculation language that you use for defining security filters. For example, the following is a valid member specification, indicating all eastern states, except "New York", for months of "Qtr1": `@REMOVE(@DESCENDANTS("Eastern Region"), "New York"), @CHILDREN(Qtr1)`.

To add a region, click **Add Region**.

6. Click **Save**, and then **OK**.